درس وکنکور پایگاه داده پیشرفته

دکتـــر رمضان عباس نـــژادورزی
مو لفین : دکتـــــر مرتضـــــی بابـــــا زاده
دکتـــر میـــر ســـعید حســـینی

برخی از عناوین مهم

تراکنش ها
تئوری پی در پی پذیری
قفل گذاری ها
زمان بندی بدون قفل
ترمیم پایگاه داده
امنیت در پایگاه داده
حل تشریحی بیش از ۲۵۰ پرسش های چهار گزینه ای
حل تشریحی بیش از ۱۰۰ مسئله پایگاه داده پیشرفته

درس و کنکور پایگاه داده پیشرفته

تألیف:

دکتر رمضان عباس نژادورزی
دکتر مرتضی بابازاده
دکتر میر سعید حسینی

فن‌آوری نوین

سرشناسه	:	عباس نژادورزی، رمضان، ۱۳۴۸ -
عنوان و نام پدیدآور		درس و کنکور پایگاه داده پیشرفته/تألیف رمضان عباس‌نـژادورزی، مرتضـی بابـازاده، میرسـ حسینی.
مشخصات نشر	:	مازندران:فن‌آوری نوین، ۱۳۹۶
مشخصات ظاهری	:	۲۵۶ص
شابک	:	۲۷۵۰۰۰ریال :۹۷۸-۶۰۰-۷۲۷۲-۰۵-۳
وضعیت فهرست نویسی	:	فیپا
موضوع		دانشگاه‌ها و مدارس عالی -- ایران -- آزمون‌ها
موضوع		Universities and colleges --Iran -- Examinations
موضوع		پایگاه‌های اطلاعاتی -- راهنمای آموزشی(عالی)
موضوع		Databases-- Study and teaching (Higher)
موضوع		پایگاه‌های اطلاعاتی-- آزمون‌ها و تمرین‌ها (عالی)
موضوع		Databases -- Examinations, questions, etc. (Higher)
موضوع		پایگاه‌های اطلاعاتی -- مسائل، تمرین‌ها و غیره (عالی)
موضوع		Problems, exercises, etc. (Higher -- Databases)
موضوع		آزمون دوره‌های تحصیلات تکمیلی -- ایران
موضوع		Iran -- Graduate Record Examination
شناسه افزوده		بابازاده، مرتضی، ۱۳۶۰ -
شناسه افزوده		حسینی شیروانی، میرسعید، ۱۳۵۷ -
رده بندی کنگره		۲۳۵۳LB
رده بندی دیویی		۳۷۸/۱۶۶۴
شماره کتابشناسی ملی		۴۸۷۳۱۴۴

www.fanavarienovin.net

فن‌آوری نوین **بابل، کدپستی ۴۷۱۶۷-۷۳۴۴۸** **تلفن: ۳۲۲۵۶۶۸۷-۰۱۱**

درس و کنکور پایگاه داده پیشرفته

تألیف: رمضان عباس نژاد ورزی- مرتضی بابا زاده- میر سعید حسینی

نوبت چاپ: چاپ اول

سال چاپ: پاییز ۱۳۹۶

شمارگان: ۲۰۰ جلد

قیمت: ۲۷۵۰۰ تومان

نام چاپخانه و صحافی:

شابک: ۳-۰۵-۷۲۷۲-۶۰۰-۹۷۸

نشانی ناشر: بابل، چهارراه نواب، کاظم‌بیگی، جنب مسجد منصور کاظم‌بیگی، طبقه همکف

طراح جلد: کانون آگهی و تبلیغات آبان (احمد فرجی)

تهران، خ اردیبهشت، نبش وحید نظری، پلاک ۱۴۲ تلفکس: ۶۶۴۰۰۱۴۴-۶۶۴۰۰۲۲۰

فهرست مطالب

فصل اول: مقدمه

۱-۱. سیستم‌های پایگاه داده ۹
۲ – ۱. تعریف و نحو تراکنش ۱۱
۱ – ۲ – ۱. خواص تراکنش ۱۳
۲-۲-۱. حالات تراکنش ۱۵
۳-۱. انواع تراکنش‌ها ۱۶
۴-۱. انواع اجرا ۱۸
۵-۱. مشکلات کنترل هم‌روندی ۱۸
۱-۵-۱. تغییرات گم‌شده ۱۹
۲-۵-۱. دست‌یابی به داده‌های تثبیت نشده ۲۰
۳-۵-۱. بازیابی ناسازگار ۲۱
۶-۱. ترمیم‌پذیری ۲۲
۷-۱. پایانه ورودی / خروجی ۲۴
۸-۱. اجتناب از سقط‌های آبشاری ۲۵
۹-۱. حفظ سازگاری ۲۹
۱۰-۱. ترتیب تراکنش‌ها ۳۰
۱۱-۱. محدودیت‌های پی‌درپی پذیری ۳۱
۱۲-۱. مدل سیستم پایگاه داده ۳۱
۱۳-۱. مدیر حافظه نهان ۳۳
۱۴-۱. مدیر ترمیم ۳۳
۱۵-۱. زمان‌بندها ۳۴
۱۶-۱. مدیر تراکنش ۳۵
۱۷-۱. ترتیب عملیات ۳۵
۱۸-۱. معماری سیستم پایگاه داده توزیع‌شده ۳۶
۱۹-۱. مسائل حل‌شده ۳۷
۲۰-۱. سؤالات چهارگزینه‌ای ۳۹
۲۱-۱. پاسخ تشریحی سؤالات چهارگزینه‌ای ۴۴

فصل دوم: تئوری پی در پی پذیری

۱-۲. سوابق ۵۰
۲-۲. تراکنش‌ها ۵۰
۳-۲. سابقه‌ها ۵۲

۲-۳-۱. سوابق پی‌درپی پذیر ۵۴
۲-۳-۲. هم‌ارزی سوابق ۵۴
۲-۴. عملیات فراتر از Read و Write ۶۱
۲-۵. هم‌ارزی دیدی ۶۲
۲-۶. پی‌درپی پذیری دیدی ۶۴
۲-۷. چند گرافی و تست پی‌درپی پذیری دیدی ۶۶
۲-۸. مسائل حل‌شده ۷۰
۲-۹. سؤالات چهارگزینه‌ای ۸۹
۲-۱۰. پاسخ تشریحی سؤالات چهارگزینه‌ای ۹۲

فصل سوم: قفل گذاری دو مرحله‌ای

۳-۱. زمان‌بندی‌های تهاجمی و محافظه‌کارانه ۹۵
۳-۲. قفل‌گذاری دومرحله‌ای پایه ۹۷
۳-۳. اثبات صحت قفل گذاری دو مرحله ای پایه (2PL پایه) ۱۰۳
۳-۴. بن‌بست‌ها ۱۰۶
۳-۵. انواع 2PL ۱۰۸
۳-۵-۱. پروتکل قفل‌گذاری دومرحله‌ای محافظه‌کار ۱۰۸
۳-۵-۲. پروتکل قفل‌گذاری دومرحله‌ای محض ۱۰۹
۳-۶. مباحث پیاده‌سازی ۱۱۱
۳-۷. مدیر قفل ۱۱۲
۳-۸. تراکنش‌های بلاک و سقط شده ۱۱۳
۳-۹. اتمیک بودن اعمال خواندن و نوشتن ۱۱۴
۳-۱۰. قفل‌گذاری عملیات اضافی ۱۱۵
۳-۱۱. صحت ۱۲۱
۳-۱۲. مباحث پیاده‌سازی ۱۲۲
۳-۱۳. مسائل حل‌شده ۱۲۳
۳-۱۴. سؤالات چهارگزینه‌ای ۱۳۱
۳-۱۵. پاسخ تشریحی سؤالات چهارگزینه‌ای ۱۳۸

فصل چهارم: زمان بندی غیر قفلی

۴-۱. مقدمه ۱۴۵
۴-۲. ترتیب مهر زمانی ۱۴۵
۴-۳. ترتیب مهر زمانی پایه‌ای ۱۴۶
۴-۴. زمان‌بند ترتیب مهر زمانی محض ۱۴۹
۴-۵. مدیریت مهر زمانی ۱۵۰
۴-۶. زمان‌بند ترتیب مهر زمانی محافظه‌کار ۱۵۱
۴-۷. زمان‌بند آزمون گراف پی‌درپی پذیری ۱۵۴
۴-۸. ملاحظات ترمیم‌پذیری ۱۵۷
۴-۹. مسائل حل‌شده ۱۵۹
۴-۱۰. سؤالات چهارگزینه‌ای ۱۶۵
۴-۱۱. پاسخ تشریحی سؤالات چهارگزینه‌ای ۱۶۷

فصل پنجم: خرابی ها

۵-۱. خرابی‌ها ۱۷۰
۵-۲. انواع حافظه‌های ذخیره‌سازی ۱۷۱
۵-۳. معماری مدیر داده ۱۷۱
۵-۴. حافظه پایدار ۱۷۲
۵-۵. مدیریت حافظه نهان ۱۷۴
۵-۶. مدیر ترمیم ۱۷۵
۵-۷. ثبت کارنامه ۱۷۶
۵-۸. اعمال UNDO و REDO ۱۷۸
۵-۹. زباله روبی ۱۷۹
۵-۱۰. همانی Restart ۱۸۰
۵-۱۱. الگوریتم REDO/ UNDO ۱۸۱
۵-۱۲. قوانین UNDO و REDO ۱۸۳
۵-۱۳. نقطه بازرسی ۱۸۴
۵-۱۴. پیاده‌سازی REDO / UNDO ۱۸۶
۵-۱۵. Restart ۱۸۹
۵-۱۶. کارنامه گیری منطقی ۱۹۴
۵-۱۷. قفل‌گذاری در سطح رکورد ۱۹۵

۵-۱۸. الگوریتم No-UNDO/REDO ۱۹۶
۵-۱۹. پیاده‌سازی ۱۹۷
۵-۲۰. الگوریتم NO-UNDO /NO-REDO ۱۹۸
۵-۲۱. خرابی‌های رسانه ۲۰۲
۵-۲۲. مسائل حل‌شده ۲۰۶
۵-۲۳. سؤالات چهارگزینه‌ای ۲۲۱۲
۵-۲۴. پاسخ سؤالات چهارگزینه‌ای ۲۱۹
فصل ششم: امنیت در پایگاه داده
۶-۱. اهمیت امنیت اطلاعات ۲۲۳
۶-۲. مفاهیم اولیه امنیت اطلاعات ۲۲۶
۶-۳. امنیت در پایگاه داده ۲۲۷
۶-۴. انواع حملات روی پایگاه داده ۲۳۰
۶-۴-۱. استنتاج ۲۳۰
۶-۴-۲. حمله غیرفعال روی پایگاه داده ۲۳۱
۶-۴-۳. حملات فعال در پایگاه داده ۲۳۲
۶-۵. SQLIA (حمله تزریق SQL) ۲۳۲
۶-۶. مکانیزم کنترل دسترسی ۲۳
۶-۷. رمزگذاری ساده ۲۳۹
۶-۸. به هم‌ریزی داده(درهم آمیختن داده) ۲۴۲
۶-۹. تکنیک‌های متفرقه برای امنیت پایگاه داده ۲۴۵
۶-۱۰. سؤالات چهارگزینه‌ای ۲۴۵
۶-۱۱. پاسخ تشریحی سؤالات چهارگزینه‌ای ۲۵۰
منابع: ۲۵۳

مقدمه

امروزه با رشد اینترنت، استفاده از داده‌ها به خصوص داده‌های عظیم روز به روز در حال افزایش می‌باشد. به همین دلیل، پایگاه داده پیشرفته به عنوان یکی از دروس در دوره کارشناسی ارشد در وزارت علوم و دانشگاه آزاد اسلامی تدریس می‌شود.

به طوری که در آزمون دکتری نرم‌افزار نیز از آن سوال می‌آید. بر همین اساس، کتاب حاضر تهیه گردید و در اختیار علاقه‌مندان قرار گرفته است. این کتاب شامل مباحث زیر می‌باشد:

۱. مفهوم تراکنش و پیاده‌سازی آن، معماری پایگاه داده، انواع زمان‌بندی با قفل گذاری و بدون قفل گذاری، ترمیم و امنیت در پایگاه داده.

۲. بیان حدود ۲۵۰ تست فراگیر و سوالات ترمی پیام نور با پاسخ تشریحی آن‌ها

۳. بیان حدود ۱۰۰ مسئله پایگاه داده پیشرفته و پاسخ تشریحی آن‌ها.

امیدوارم این اثر مورد توجه جامعه انفورماتیک کشور، اساتید و دانشجویان عزیز قرار گیرد.

در پایان از تمامی اساتید و دانشجویان عزیز تقاضا داریم، هر گونه اشکال، ابهام در متن کتاب، پیشنهاد و انتقادات را به آدرس پست‌الکترونیک **fanavarienovin@gmail.com** ارسال نمایند.

مولفین

fanavarienovin@gmail.com

فصل ۱ مقدمه

کنترل هم‌روندی فعالیتی است که رفتار پردازش‌هایی را که به‌طور موازی عمل می‌کنند، هماهنگ می‌سازد، دسترسی داده را به اشتراک می‌گذارد و بنابراین به‌طور بالقوه تراکنش‌ها را با یکدیگر قاطی می‌کند. **ترمیم**[1]، فعالیتی است که تضمین می‌کند خرابی‌ها سخت‌افزار و نرم‌افزار باعث تخریب ماندگاری داده نمی‌شود. کنترل هم‌روندی و مشکلات ترمیم در طراحی سخت‌افزار، سیستم‌های عامل، سیستم‌های بلادرنگ، سیستم‌های ارتباطی، سیستم‌های پایگاه و غیره به وجود می‌آیند. در این کتاب درباره کنترل هم‌روندی و مشکلات ترمیم در سیستم‌های پایگاه داده بحث می‌کنیم.

با استفاده از یک مدل سیستم‌های پایگاه داده را بررسی می‌کنیم. این مدل انتزاعی از انواع سیستم‌های اداره‌کننده داده از قبیل سیستم‌های مدیریت پایگاه داده برای برنامه‌های کاربردی پردازش داده، سیستم‌های پردازش تراکنش برای رزرو خطوط هوایی یا بانکداری و سیستم‌های فایل برای محیط محاسبه عمومی است. کنترل هم‌روندی و ترمیم در هر سیستم که با مدل مطابقت دارد، اعمال می‌شود. قطعه اصلی این مدل **تراکنش**[2] است. به‌طور غیررسمی یک تراکنش اجرایی از یک برنامه است که به پایگاه داده به اشتراک گذاشته‌شده دسترسی دارد. هدف از کنترل هم‌روندی و ترمیم این است که تضمین کند، تراکنش‌ها به‌طور **اتمیک**[3] اجرا می‌شوند. یعنی:

۱. هر تراکنش به داده به اشتراک گذاشته‌شده، بدون تداخل با بقیه تراکنش‌ها دسترسی دارد.

۲. اگر یک تراکنش به‌طورمعمول پایان پذیرد، پس از اتمام آن، اثرات آن به‌صورت **پایدار (دائمی)**[4] می‌گردد.

هدف این فصل این است که به این مدل به‌طور جامع و دقیق بپردازیم. در این بخش، **مدل مبتنی برکاربر**[5] از سیستم را نشان می‌دهیم. این مدل، پایگاه داده‌ی را شامل می‌شود که کاربر بتواند به‌وسیله اجرای تراکنش‌ها به آن دسترسی داشته باشد و باوجود خرابی‌ها، تراکنش در آن به‌صورت اتمیک اجرا می‌شود. در ادامه این فصل، یک مدل از سیستم‌های کنترل هم‌روندی پایگاه داده و ترمیم ارائه‌شده است که هدف آن درک یکپارچگی تراکنش است.

۱-۱. سیستم‌های پایگاه داده

یک پایگاه داده شامل مجموعه‌ی از اقلام داده نام‌گذاری شده است. هر قلم داده دارای یک مقدار است. مقادیر اقلام داده‌ای در هر زمان شامل وضعیت پایگاه داده است. در عمل یک‌قلم داده باید یک بخش حافظه

---

[1]. Recovery [2]. Transaction [3]. Atomically [4]. Persistent [5]. user-oriented model
[6]. granularity

اصلی مانند یک صفحه دیسک، یک رکورد از یک فایل، یا یک فیلد از رکورد باشد. اندازه داده موجود در قلم داده، **دانه‌بندی**[۱] قلم داده نامیده می‌شود.

دانه‌بندی ممکن است برای بررسی‌هایمان مهم نباشد و بنابراین آن را به‌طور نامشخص باقی می‌گذاریم. وقتی‌که دانه‌بندی را به‌طور نامشخص باقی می‌گذاریم، اقلام داده‌ی را با حروف کوچک و معمولاً x ، y و z نشان می‌دهیم.

یک سیستم پایگاه داده (DBS[۲])، کلکسیونی از سخت‌افزارها و ماژول‌های نرم‌افزاری است که مجموعه‌ای از دستورات دسترسی به پایگاه داده که **اعمال پایگاه داده** [۲] (به‌طور ساده اعمال) نامیده می‌شود را پشتیبانی می‌کند. مهم‌ترین عملیاتی که پایگاه داده انجام می‌دهد Read (خواندن) و Write (نوشتن) هستند. این اعمال به‌صورت زیر به کار می‌روند:

۱. **Read(x)**، مقدار ذخیره‌شده در قلم داده‌ی x را برمی‌گرداند.

۲. **Write(x, Val)**، مقدار x را به Val تغییر می‌دهد.

همچنین پایگاه داده عملیات دیگری را انجام می‌دهد که در ادامه می‌بینیم.

سیستم پایگاه داده (DBS) هر عمل را به‌طور **اتمیک**[۳] (یکپارچه) اجرا می‌کند. یعنی، سیستم پایگاه داده اعمال را به‌طور **پی‌درپی**[۴] (**به ترتیب**) اجرا می‌کند. پس، هر عمل را در یک‌زمان انجام می‌دهد. برای به دست آوردن این رفتار سیستم پایگاه داده ممکن است واقعاً اعمال را به‌صورت پی‌درپی اجرا کند. هرچند، حتی اگر عملیات در سیستم پایگاه داده به‌صورت **هم‌روند**[۵] اجرا شوند، اثر نهایی آن‌ها باید معادل اجرای پی‌درپی آن اعمال باشد. به‌عنوان‌مثال، فرض کنید اقلام داده‌ی x و y در دو دیسک مختلف ذخیره‌شده‌اند. سیستم پایگاه داده ممکن است اعمال روی x و y را به ترتیب زیر اجرا کند:

۱. اجرای Read(x) را انجام دهد.

۲. پس از اجرای Read(x) (خاتمه گام ۱)، به‌صورت هم‌روند یا موازی اجرای Write(x, 1) و Read(y) را انجام دهد.

۳. پس از اجرای گام ۲، اجرای Write(y, 0) را انجام دهد.

هرچند Write(x, 1) و Read(y) به‌طور هم‌روند اجرا می‌شوند، می‌توان فرض کرد که به‌طور اتمیک اجراشده‌اند. چون این اجرا همان اثری را دارد که به‌طور پی‌درپی به‌صورت دستورات زیر اجرا شوند:

Read(x), Write(x, 1), Read(y), Write(y, 0)

همچنین سیستم پایگاه داده **اعمال دیگر تراکنش**[۶] از قبیل **Start**، **Commit** و **Abort** را پشتیبانی می‌کند. **عمل Start**، شروع اجرای تراکنش جدیدی را به سیستم پایگاه داده اعلان می‌کند، **عمل Commit** یا **Abort** یک تراکنش را خاتمه می‌دهد. عمل **Commit**، خاتمه نرمال (تثبیت) تراکنش را به سیستم پایگاه داده اعلان می‌نماید و تمام اثرات تراکنش را دائمی می‌کند. **عمل Abort**، بیان می‌کند که تراکنش با خطا مواجه شده است و تمام اثرات آن باید نادیده گرفته شود.

---

[2].Database System(DBS) [2]. Database Operations [3].Atomicily [4]. Sequentially
[5]. Concurrently [6]. Transaction Operations

به ازای هر تراکنش خاص، باید مشخص گردد که کدام برنامه دستورات پایگاه داده مربوطه را صادر نموده است. برای این منظور، در این مدل، فرض می‌شود که سیستم پایگاه داده به ازای هر Start یک **شناسه‌ی منحصربه‌فرد (یکتا)** [۳] به تراکنش تخصیص می‌دهد. برنامه این شناسه را به هر یک از اعمال پایگاه داده خودش ضمیمه می‌کند تا زمانی که تراکنش Commit یا Abort شود. در نتیجه، از نظر سیستم پایگاه داده یک تراکنش با عمل Start شروع می‌شود، با تعدادی از اعمال دیگر پایگاه داده (احتمالاً هم‌روند) ادامه می‌یابد و در نهایت با یک دستور Commit یا Abort خاتمه می‌یابد. یک تراکنش ممکن است اجرای هم‌روند دو یا چند برنامه باشد. یعنی، در تراکنش ممکن است که قبل از این‌که نتیجه عمل اول مشخص باشد، عمل دوم (بعدی) ارائه گردد. هرچند آخرین عمل تراکنش‌ها باید Commit یا Abort باشد. بنابراین، یک سیستم پایگاه داده باید از پردازش مجدد داده‌های یک تراکنش که پس از اعمال Commit یا Abort رسیده‌اند، اجتناب نماید.

۲ – ۱. تعریف و نحو تراکنش

کاربران معمولاً از طریق اجرای برنامه‌ها با سیستم پایگاه داده ارتباط برقرار می‌کنند. از دیدگاه کاربر، یک تراکنش اجرای یک یا چند برنامه است که شامل اعمال پایگاه داده‌ی است. در پایگاه داده، تراکنش حاوی مجموعه‌ای از یک یا چند پرس‌وجو است که به‌عنوان یک واحد عمل می‌کنند. نتیجه اجرای هر پرس‌وجو در این واحد به **تثبیت**[۲] یا **سقط**[۳] دستورات دیگر بستگی دارد. یعنی، این واحد به‌عنوان یک کل، قابل‌تفکیک نیست. اگر حداقل یکی از پرس‌وجوها در این واحد با موفقیت اجرا نشود، کل آن واحد **خنثی**[۴] خواهد شد و داده‌هایی که در اجرای آن واحد تغییر کرده‌اند، به مقدار قبل از اجرای تراکنش (آخرین مقدار تثبیت‌شده)[۵] برمی‌گردند. هنگامی تراکنشی با موفقیت تثبیت خواهد شد که همه دستورات آن تثبیت گردند.

دنیای واقعی پر از تراکنش‌هایی مانند تراکنش‌ها در بورس، خریدهای اینترنتی از طریق کارت، کنترل موجودی حساب‌های بانکی و غیره است. در همه این موارد، موفقیت اجرای تراکنش به تعدادی دستورات بستگی دارد که به هم وابسته هستند. شکست هر یک از آن دستورات، تراکنش را سقط کرده و سیستم را به حالت قبل از اجرای تراکنش برمی‌گرداند.

برای درک بهتر مفهوم تراکنش، مثال انتقال پول از حساب A به‌حساب B را در نظر بگیرید (شکل ۱–۱). در این شکل می‌خواهیم ۱۰۰۰۰۰۰ ریال از حساب A به‌حساب B منتقل شود. در این سیستم زمانی انتقال کامل می‌گردد که حساب A، ۱۰۰۰۰۰۰ ریال بدهکار و حساب B، ۱۰۰۰۰۰۰ ریال بستانکار شود. اگر هر یک از این مراحل با شکست مواجه شود (انجام نگردد)، انتقال پول انجام نمی‌شود و هیچ مبلغی نباید از حساب A برداشت گردد. بنابراین، انتقال پول از حساب A به‌حساب B را می‌توان یک تراکنش فرض کرد. یعنی، یک واحد کار مستقل که شامل دو دستور است. این دو دستور عبارت‌اند از:

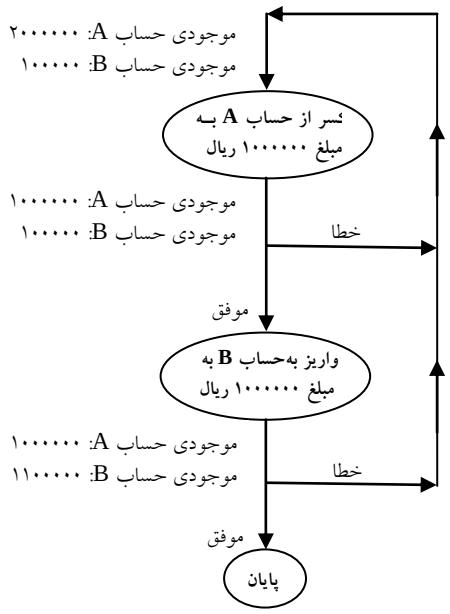
۱. برداشت ۱۰۰۰۰۰۰ ریال از حساب A

۲. واریز ۱۰۰۰۰۰۰ ریال به‌حساب B

---

[3]. Unique Transaction identifier [2]. Commit [3]. Abort [4]. Undo [5]. Last Committed Value

در صورتی این تراکنش با موفقیت اجرا خواهد شد که هر دو کار(برداشت از حساب A و واریز به‌حساب B) با موفقیت تثبیت گردند. اگر یکی از این کارها با شکست مواجه شود، تراکنش سقط می‌گردد و پایگاه داده به وضعیت پایدار قبل از شروع این تراکنش برمی‌گردد.



شکل ۱ - ۱ تراکنش انتقال ۱۰۰۰۰۰۰ ریال از حساب A به‌حساب B.

یک پایگاه داده بانکداری را در نظر بگیرید که فایل Accounts شماره‌حساب مشتریان را نگه‌داری می‌کند. در این فایل برای هر مشتری مانده‌حسابش نگه‌داری می‌شود و شماره‌حساب کلید آن در فایل Accounts است. یک تراکنش نمونه برای انتقال پول از یک حساب به‌حساب دیگر رویه‌ای به‌صورت زیر است (پیاده‌سازی تراکنش شکل ۱-۱):

```
Procedure Transfer
begin
  Start;
  input(fromAccount, toAccount, amount);
  (* This procedure transfers "amount" from "fromAccount" into "toAccount." "*)
  temp : = Read(Accounts[fromAccount]);
  if temp < amount then
   begin
    output( "موجودی کافی نیست");
    Abort
  end
  else
  begin
    Write(Accounts[fromAccount], temp - amount);
     temp : = Read(Accounts[toAccount]);
    Write(Accounts[toAccount], temp + amount);
```

```
    Commit;
    output( ''عمل انتقال کامل شد");
end;
return
End
```

۱۴-۱. مدیر ترمیم

مهم‌ترین مسئولیت مدیر ترمیم (RM=Recovery Manager) این است که تضمین کند که پایگاه داده شامل اثرات تراکنش‌های تثبیت‌شده و فاقد اثرات تراکنش‌های سقط شده است. به همین دلیل، مدیر ترمیم از عملیات Start، Commit، Abort، Read و Write پشتیبانی می‌کند. این عملیات را با استفاده از عملیات Fetch و Flush مربوط مدیر حافظه نهان پردازش می‌کند. به‌طورمعمول، ترمیم طوری طراحی می‌شود تا نسبت به خطاهای که موجب از دست رفتن محتوی حافظه فرار می‌شود، مقاوم باشد. چنین خطاهای ''**خطاهای سیستمی**''[۳] نامیده می‌شوند.

بعد از ترمیم سیستم از خطای سیستمی، مدیر ترمیم باید ضمانت دهد که پایگاه داده شامل تمام اثرات تراکنش‌های تثبیت‌شده و فاقد اثرات تراکنش‌های که سقط شده(در زمان خطا فعال بوده‌اند) است. مدیر ترمیم باید اثرات تراکنش‌هایی که در هنگام خطا فعال بوده‌اند را از بین ببرد، چون این تراکنش‌ها وضعیت داخلی‌شان را به دلیل از دست دادن محتوی حافظه اصلی ازدست‌داده‌اند و بنابراین نمی‌توانند اجرای خود را تثبیت کنند.

پس از اتفاق افتادن یک خطا سیستمی، تنها اطلاعاتی که مدیر ترمیم در دسترس دارد، محتویات حافظه پایدار است. چون مدیر ترمیم نمی‌داند که چه زمانی یک خطایی سیستمی رخ می‌دهد، مدیر ترمیم باید نسبت به انتقال داده بین حافظه فرار و حافظه پایدار خیلی دقیق باشد. در غیر این صورت، پس از خطای سیستمی ممکن است سیستم دریکی از دو وضعیت ترمیم‌ناپذیر زیر قرار گیرد:

۱. به‌روزرسانی بعضی از تراکنش‌های تثبیت‌شده، در حافظه پایدار موجود نباشد.

۲. حافظه پایدار شامل مقدار x که توسط تراکنش تثبیت نشده‌ای نوشته‌شده باشد، است. اما، آخرین مقدار x توسط تراکنش تثبیت‌شده‌ای نوشته‌شده است را شامل نباشد.

برای اجتناب از این مشکلات، مدیر ترمیم ممکن است لازم بداند که خود را به حالاتی محدود نماید که مدیر حافظه نهان بتواند یک‌طرفه تصمیم به اجرای یک عمل Flush را بگیرد. مدیر ترمیم همچنین می‌تواند به‌گونه‌ای طراحی شود تا نسبت به خطاهای بخش‌هایی از حافظه پایدار مقاوم باشد. خطاهای بخش‌هایی از حافظه پایدار، **خطاهای رسانه**[۴] نامیده می‌شوند. برای انجام چنین کاری، لازم است تا کپی‌های از داده روی حداقل دو دستگاه حافظه پایدار متفاوت نگه‌داری شود، به‌طوری احتمال خطای هم‌زمان این حافظه‌ها کم باشد. در برخورد با خطاهای رسانه لازم است تا مدیر ترمیم قادر باشد پایگاه داده را به وضعیتی برگرداند که شامل اثر تمام تراکنش‌های به‌روزرسانی تثبیت‌شده باشد و عاری از اثر به‌روزرسانی تراکنش‌های تثبیت نشده نیز باشد.

۱۵-۱. زمان‌بندها

---

[4]. Media Failures [2].Execute [3].Reject [4].Defer [5].Delay

زمان‌بند، مجموعه‌ای از برنامه‌هاست که اجرای هم‌روند تراکنش‌ها را کنترل می‌کند. برای انجام این کنترل، زمان‌بند ترتیبی اتخاذ می‌دهد که مدیر داده عملیات Readها، Commitها، Writeها و Abortهای تراکنش‌های مختلف را انجام می‌دهد. هدف این است که ترتیب عملیات طوری باشد که نتیجه اجرا حداقل پی‌درپی پذیر و ترمیم‌پذیر باشد که بسته به نیاز زمان‌بند تضمین می‌کند که اجرا فاقد سقط آبشاری یا محض باشد.

جهت اجرای یک عملیات پایگاه داده، تراکنش عمل را به زمان‌بند ارسال می‌کند. زمان‌بند پس از دریافت عمل، می‌تواند سه عمل زیر را انجام دهد:

۱. **اجرا**[۲]، زمان‌بند عمل را به مدیر داده ارسال می‌کند. وقتی مدیر داده اجرای عمل را خاتمه داد، به زمان‌بند اطلاع می‌دهد. به‌علاوه، اگر عمل درخواست Read باشد، مدیر داده مقدار (مقادیر خوانده خودش) را به زمان‌بند برمی‌گرداند.

۲. **رد**[۳]، از اجرای عمل اجتناب می‌کند، به تراکنش‌ها می‌گوید عملیات‌شان رد شده است. این عمل موجب می‌شود تا تراکنش‌ها سقط شوند. عمل سقط می‌تواند توسط خود تراکنش یا مدیر تراکنش انجام شود.

۳. **تعویق**[۴] **یا تأخیر**[۵]، زمان‌بند با قرار دادن عمل در یک صف داخلی، آن را به تعویق می‌اندازد. بعداً زمان‌بند عمل را از صف حذف کرده، آن را اجرا می‌کند، یا رد می‌نماید (بدون اجرا آن را دور می‌اندازد). زمان‌بند، در مدت تأخیر عمل ممکن است تصمیم بگیرد که اعمال دیگری را زمان‌بندی کند.

زمان‌بند با استفاده از سه عکس‌العمل (اجرای یک عمل، رد کردن یا به تأخیر انداختن آن)، می‌تواند ترتیب انجام عملیات را کنترل نماید. وقتی‌که زمان‌بند عمل را از مدیر تراکنش دریافت می‌کند، معمولاً سعی می‌کند آن را به بلافاصله به مدیر داده بفرستد. اگر زمان‌بند بتواند بدون آن‌که اجرای پی‌درپی پذیری را تولید کند آن را انجام دهد و اگر تصمیم بگیرد عمل را اجرا کند، ممکن است نتیجه‌ی نادرستی را تولید کند. پس یا آن عمل را به تأخیر می‌اندازد (اگر توانایی آن را داشته باشد که در آینده به‌طور صحیح عمل را پردازش کند) یا عمل را رد کند(هیچ‌گاه توانایی آن را ندارد که عمل را در آینده به‌طور صحیح پردازش کند). بنابراین، زمان‌بند از تعویق و رد عمل استفاده می‌کند تا به تولید صحیح اجراها کمک کند. به‌عنوان‌مثال، اجازه دهید اجرای هم‌روند و تراکنش Deposit که مبالغ ۱۱۰۰ و ۱۰۰۰۰۰ ریال را به‌حساب شماره ۱۳ واریز می‌کنند، در نظر بگیریم:

$Read_1$ (Accounts [13]);
$Read_2$ (Accounts [13]);
$Write_2$ (Accounts [13], 101000);
$Commit_2$;
$Write_1$ (Accounts [13], 1100);
$Commit_1$.

یک زمان‌بند برای اجتناب از این اجرای پی‌درپی ناپذیر ممکن است تصمیم بگیرد که $Write_1$ را رد کند، بنابراین تراکنش T_1 را سقط می‌کند. در این حالت، کاربر یا مدیر تراکنش می‌تواند دوباره تراکنش T_1 را صادر کند تا بدون هیچ‌گونه تداخل با T_2 انجام شود. به‌طور مشابه، زمان‌بند می‌توانست $Read_2$ تا زمانی که $Write_1$ دریافت و اجرا شود، به تأخیر بی‌اندازد. زمان‌بند با به تعویق انداختن $Read_2$، از انجام رد $Write_1$ در ادامه جلوگیری می‌کند.

۱۹–۱. مسائل حل‌شده

مثال ۱. کدام‌یک از زمان‌بندی‌های زیر مشکل هم‌روندی دارند؟

1) $Read_1(x), Write_1(x), Read_2(x), Write_2(y), Abort_1, Commit_2$
2) $Read_1(x), Write_1(x), Read_2(y), Write_2(y), Abort_1, Commit_2$
3) $Read_1(x), Read_2(x), Read_2(y), Write_2(y), Read_1(z), Abort_1, Commit_2$
4) $Read_1(x), Read_2(x), Write_2(x), Write_1(x), Commit_1, Commit_2$
5) $Read_1(x), Read_2(x), Write_2(x), Read_1(y), Commit_1, Commit_2$
6) $Read_1(x), Write_1(x), Read_2(x), Write_2(x), Commit_1, Commit_2$

گزینه (۱) یک حالت خواندن کثیف (Dirty Read) است. چون عمل $Read_2(x)$ مقدار نوشته‌شده به‌وسیله $Write_1(x)$ را می‌خواند. پس تراکنش T_1، Abort شده است. گزینه (۲)، هیچ مشکلی ندارد. چون دو تراکنش به دو داده متفاوت دسترسی دارند (تراکنش T_1 به x و تراکنش T_2 به y دسترسی دارد). گزینه (۳)، هیچ مشکلی ندارد، چون تراکنش T_1 روی هیچ داده‌ای نمی‌نویسد. گزینه (۴)، مشکل تغییرات گم‌شده را دارد، چون تراکنش T_1 روی‌داده x که تراکنش T_2 نوشته است، می‌نویسد. گزینه‌های (۵) و (۶) نیز فاقد هرگونه مشکل هستند.

مثال ۲. سه تراکنش زیر را در نظر بگیرید:

T_1: R(x), W(x), R(y), W(y), C_1
T_2: W(x), R(y), R(z), C_2
T_3: W(x), R(y), C_3

الف: زمان‌بندی برای این تراکنش‌ها بنویسید که ترمیم‌پذیر باشد، اما، از سقط آبشاری اجتناب نکند؟
ب: زمان‌بندی برای این تراکنش‌ها بنویسید که ترمیم‌پذیر نباشد؟

الف: زمان‌بندی زیر را در نظر بگیرید:

زمان/تراکنش	1	2	3	4	5	6	7	8	9	10	11	12
T_1			R(x)	W(x)						R(y)	W(y)	C_1
T_2	W(x)	R(y)			R(z)	C_2						
T_3							W(x)	R(y)	C_3			

این زمان‌بندی ترمیم‌پذیر است، چون تراکنش T_1 مقدار x را از تراکنش T_2 خوانده است و C_2 قبل از C_1 قرار دارد. ولی از سقط آبشاری اجتناب نمی‌کند، چون تراکنش T_1 مقدار قلم داده‌ای x را قبل از تثبیت تراکنش T_2 از آن خوانده است.

ب : اکنون زمان‌بندی زیر را مشاهده کنید:

زمان/تراکنش	1	2	3	4	5	6	7	8	9	10	11	12
T_1			R(x)	W(x)	C_1						R(y)	W(y)
T_2	W(x)	R(y)				R(z)	C_2					
T_3								W(x)	R(y)	C3		

این زمان‌بندی ترمیم‌پذیر نیست، چون تراکنش T1 مقدار قلم داده x را از تراکنش T2 خوانده است. اما، خودش زودتر تثبیت کرده است. برای ترمیم‌پذیر بودن ابتدا باید T2 و سپس T1 تثبیت کند.

۲۰–۱. سؤالات چهارگزینه‌ای

۱. عبارت زیر مربوط به کدام خصوصیت پایگاه داده‌ها می‌باشد؟
« تا زمانی که تراکنش هنوز Commit نشده باشد نباید اطلاعاتش را در اختیار تراکنش‌های دیگر قرار دهد.»

الف: Atomicity ب: Consistency ج: Durability د: Isolation

۲. عبارت زیر مربوط به کدام خصوصیت پایگاه داده‌ها می‌باشد؟
« تراکنش‌ها بانک اطلاعاتی را از یک حالت درست به حالت درست دیگری می‌برند. »

الف: Atomicity ب: Correctness ج: Durability د: Isolation

۳. تراکنش یک واحد منطقی از کار است که با اجرای عمل BEGIN TRANSACTION شروع و با اجرای..... خاتمه پیدا می‌کند؟

الف: ROLLBACK ب: COMMIT

ج: COMMIT یا ROLLBACK د: COMMIT و ROLLBACK

٤. گزینه درست را انتخاب کنید(کدام گزینه خواص تراکنش است)؟
الف: اتمیک ب: صحت ج: وابستگی د: پایداری

الف: (الف، ب، ج، د) ب: (ب، ج، د) ج: (الف، ج، د) د: (الف، ب، د)

۵. کدام گزینه در مورد تراکنش (Transaction) بانک اطلاعاتی درست است؟

الف: هر تراکنش یک برنامه است. ب: هر تراکنش یک دستور SQL است.

ج: تراکنش واحد کار (از دید کاربر) روی بانک اطلاعاتی است.

د: هر تراکنش فقط می‌تواند یک جدول را در بانک اطلاعاتی تغییر دهد.

۲۱-۱. پاسخ تشریحی سؤالات چهارگزینه‌ای

۱. **گزینه (د) صحیح است.** گزینه (الف)، یکپارچگی یا اتمیک (Atomicity) تضمین می‌کند که یا همه دستورات تراکنش اجرا می‌شوند یا هیچ‌کدام اجرا نمی‌شوند، گزینه (ب)، سازگاری یا صحت (Consistency) بیان می‌کند که اولاً بانک اطلاعاتی قبل از انجام تراکنش در یک حالت صحیح قرار دارد و ثانیاً پس از انجام تراکنش به حالت صحیح دیگری می‌رود، گزینه (ج)، پایداری، پایایی یا ماندگاری (Durability) تضمین می‌کند که اگر تراکنشی تثبیت گردید، اثر آن ماندنی بوده و هرگز به‌صورت تصادفی نابود نمی‌شود. گزینه (د)، انزوا (Isolation)، یعنی، اگر چندین تراکنش به‌طور هم‌زمان اجرا گردند، هر تراکنش باید از اجرای تراکنش‌های هم‌زمان بی‌اطلاع باشد و نتیجه یک تراکنش تأثیر مخرب بر روی تراکنش دیگر نداشته باشد. اگر T_i و T_j دو تراکنش باشند، قبل از شروع تراکنش T_j باید اجرای T_i تمام شده باشد.

۲. **گزینه (ب) صحیح است.** خاصیت یکپارچگی (Atomicity)، یا همه‌ی دستورات تراکنش باید انجام شوند، یا هیچ کدام اجرا نشود. خاصیت انزوا (Isolation)، بیان می‌کند که تراکنش‌های هم‌روند نباید تأثیر مخرب روی هم داشته باشند. خاصیت پایایی (Durability)، بیان می‌کند که اگر تراکنش تثبیت (COMMIT) گردید، تأثیر آن به‌طور اتفاقی از بین نخواهد رفت.

۳. **گزینه (ج) صحیح است.** گزینه (د)، نادرست است، چون بیان کرده که با اجرای ROLLBACK یا COMMIT تراکنش خاتمه پیدا می‌کند که آخرین دستور تراکنش می‌تواند COMMIT (تثبیت و پذیرش تغییر تراکنش) یا ROLLBACK (خنثی یا لغو نمودن تأثیر دستورات تراکنش) باشد.

۴. **گزینه (د) صحیح است.** چون خواص تراکنش اتمیک (Atomicity)، انزوا (Isolation)، پایایی یا پایداری (Durability) و صحت (سازگاری یا Consistency) هستند.

۵. گزینه (ج) صحیح است. گزینه (الف)، نادرست است، چون تراکنش یک برنامه است که یک کار واحد را روی بانک اطلاعاتی انجام می‌دهد، گزینه (ب)، نادرست است، چون یک تراکنش ممکن است چندین دستور باشد، گزینه (د)، نیز نادرست است، چون یک تراکنش می‌تواند چندین جدول را تغییر دهد.

فصل ۲ تئوری پی‌درپی پذیری

۲-۱. سوابق

تئوری پی‌درپی پذیری ابزاری ریاضی جهت اثبات صحت کار یک زمان‌بند است. در این تئوری اجرای هم‌روند مجموعه‌ای از تراکنش‌ها توسط ساختاری به نام **سابقه**[5] نمایش داده می‌شود. اگر سابقه‌ای یک اجرای پی‌درپی را ارائه کند، یک **سابقه پی‌درپی پذیر**[2] نامیده می‌شود. تئوری پی‌درپی پذیری خواص دقیق‌تر و مختصرتری در رابطه با سابقه ارائه می‌نماید تا پی‌درپی پذیری را برآورده کند.

۲-۲. تراکنش‌ها

توسعه تئوری پی‌درپی پذیری را به‌وسیله توصیف این‌که چگونه تراکنش‌ها مدل می‌شوند، شروع می‌کنیم. همان‌طور که در فصل اول بیان گردید، تراکنش، اجرای خاصی از برنامه‌ی است که پایگاه داده را با عملیات Read و Write دست‌کاری می‌کند. ازنقطه‌نظر تئوری پی‌درپی پذیری، یک تراکنش ارائه دوباره از یک اجرا است که در آن عملیات Read ، Write و ترتیب آن‌ها بیان‌شده باشد. تراکنش برای هر عمل Read و Write فقط نام قلم داده را مشخص می‌کند، و کاری به مقدار خوانده‌شده یا نوشته‌شده ندارد. به‌علاوه، هر تراکنش همچنین شامل عملیات Commit یا Abort است. Commit، تعیین می‌کند که تراکنش با موفقیت خاتمه یافته است. اما، Abort مشخص می‌کند که تراکنش با خطا متوقف‌شده است.

به‌عنوان‌مثال، اجرای برنامه زیر را ببینید:

```
Procedure P
begin
  Start;
  temp : = Read(x);
  temp : = temp + 1;
  Write(x, temp);
  Commit
End
```

که ممکن است به‌صورت زیر نمایش داده شود:

$r_1[X] \rightarrow w_1[X] \rightarrow c_1$

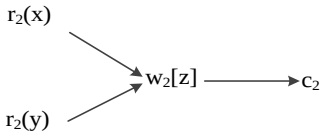
اندیس، شماره تراکنش را مشخص می‌کند. اندیس برای تمایز عملیات یک تراکنش از عملیات تراکنش‌های دیگر (که برای دسترسی به اقلام داده و ترتیب اتفاق افتادن آن‌ها) به کار می‌رود.

به‌طورکلی $r_i[X]$ (یا $w_i[X]$) برای معرفی عمل خواندن (یا نوشتن) توسط تراکنش T_i روی قلم داده X استفاده می‌شود. در این نشانه‌گذاری فرض می‌کنیم که هیچ تراکنشی قلم داده را بیش‌تر از یک‌بار نخوانده یا ننوشته است. دستورات c_i و a_i به ترتیب برای عملیات Commit و Abort تراکنش T_i استفاده می‌شوند.

---

[5]. History [2]. Serializable

در هر تراکنش، فقط یکی از دستورات a_i یا c_i ظاهر می‌شوند. علامت → ترتیب اجرای عملیات را نشان می‌دهد. بنابراین، در مثال ($r_1[X] \rightarrow w_1[X] \rightarrow c_1$)، علامت → بیان می‌کند که $w_1[X]$ بعد از $r_1[x]$ و **قبل از** c_1 اتفاق می‌افتد. همچنین، در فصل اول دیدیم، از اجرای هم‌روند (موازی) تراکنش‌ها، برنامه‌ها تولید می‌شوند. به‌عنوان‌مثال، برنامه‌ی که به‌طور موازی اقلام داده‌ی x و y را می‌خواند و مجموع آن‌ها را در z می‌نویسد، ممکن است دو خواندن را به‌صورت موازی انجام دهد. این نوع از اجرا به‌عنوان یک **ترتیب جزئی**[6] مدل می‌شود. یعنی، مجموعه‌ای که برخی یا تعدادی از اعضای آن ترتیب مشخصی دارند. به عنوان مثال، اجرای موجود در شکل ۱-۲ را ببینید:



شکل ۱-۲ نمایش اجرای هم‌روندی

این اجرا می‌گوید که عمل $w_2[z]$ باید بعد از انجام دو عمل $r_2[x]$ و $r_2[y]$ اتفاق بی افتد، اما ترتیب دو عمل $r_2[x]$ و $r_2[y]$ مهم نیست. اگر تراکنش هم عمل خواندن و هم عمل نوشتن را روی‌داده x انجام می‌دهد، به ترتیب جزئی خاص **ترتیب نسبی**[2] از Read(x) و Write(x) نیاز داریم. این بدین دلیل است که ترتیب اجرای عملیات درنتیجه عملیات تأثیر می‌گذارد. یعنی، مقدار x که برگشت داده می‌شود، به این‌که آیا این عملیات قبل یا بعدازآن Write(x) بوده است، بستگی دارد.

اما، اگر در یک مجموعه تمام اعضا ترتیب مشخصی داشته باشند، **ترتیب کلی**[3] نام دارد.

می‌خواهیم تعریف یک تراکنش را به‌صورت ترتیب جزئی عملیات فرمول نماییم. در ریاضی، یک ترتیب جزئی با زوج مرتب ($\sum$ و $<$) نشان داده می‌شود. نماد $\sum$، نشان‌دهنده مجموعه‌ی از عناصری است که می‌خواهند مرتب شوند و < رابطه ترتیب آن عناصر را تعیین می‌کند. بر این اساس، هر تراکنش T_i یک زوج مرتب ($\sum_i$ و $<_i$) است. نماد $\sum_i$، نشان‌دهنده مجموعه عملیات و نماد $<_i$ معرف ترتیب اجرای آن عملیات است.

این روش نمایش، کمی پیچیده است. به همین دلیل، به‌جای نماد $\sum$ می‌توانیم از نام ترتیب جزئی استفاده کنیم. در این حالت T_i هم برای نامیدن ترتیب جزئی و هم مجموعه عناصر (یعنی، عملیات) در ترتیب جزئی استفاده می‌شود. وقتی‌که می‌نویسیم $r_i[x] \in T_i$ یعنی، $r_i[x]$ یک عضو [عملیات] از T_i است. T_i برای نام‌گذاری مجموعه عملیات در ترتیب جزئی استفاده می‌شود. تعریف رسمی تراکنش در زیر آمده است:

تراکنش T_iیک ترتیب جزئی با رابطه ترتیب $<_i$ است که در آن:

۱. $T_i \sqsubseteq \{r_i[x], w_i[x] \cup \{a_i, c_i\}$ **است که x یک قلم داده است.** این شرط، نوع عملیات در تراکنش را تعریف می‌کند. این عملیات شامل مجموعه خواندن و نوشتن داده‌ها، تثبیت یا سقط است.

۲. $a_i \in T_i$ **است، اگر و فقط اگر** $c_i \in T_i$ **باشد.** این شرط بیان می‌کند که تراکنش شامل تنها یک Commit یا یک Abort (نه هردوی آن‌ها) است.

---

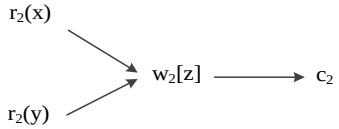
6. Partial order 2. Relative order 3. Total order

۳. **اگر t یکی از عملیات c_i یا a_i باشد، برای هر عمل دیگر $p \in T_i$، $t <_i p$ است.** این شرط بیان می‌کند که $<_i$ ترتیب عملیات است و Commit و Abort باید بعد از همه عملیات بی‌آیند. یعنی، Commit یا Abort در انتهای(آخر) تراکنش قرار می‌گیرند.

۴. **اگر $ri[x], wi[x] \in Ti$ باشند، پس، یکی از رابطه‌های $w_i(x) <_i r_i[x]$ یا $r_i[x] <_i w_i(x)$ برقرار است.** در این شرط، $<_i$ ترتیب اجرای عملیات Read و Write را روی قلم داده مشترک مشخص می‌کند. یعنی، ترتیب خواندن و نوشتن تراکنش روی یک داده باید مشخص‌شده باشد.

برای درک ارتباط بین دو علامت به مثال زیر دقت کنید:

مثال ۱-۲. تراکنش T_2 را به صورت زیر در نظر بگیرید و به صورت رسمی آن را بیان کنید؟



به‌طور رسمی، این تراکنش بیان می‌کند که T_2 شامل مجموعه‌ی از عملیات زیر است:

$T_2 = \{r_2[x], r_2[y], w_2[z], c_2\}$
$<_2 = \{(r_2[x], w_2[z]), (r_2[y], w_2[z]), (w_2[z], c_2), (r_2[x], c_2) (r_2[y], c_2)\}$

توجه کنید که پیکان‌هایی که از طریق تعدی تولید می‌شوند، را صراحتاً منظور نمی‌کنیم. به‌عنوان‌مثال، پیکان $r_2[x] \rightarrow c_2$ توسط $r_2[x] \rightarrow w_2[z]$ و $w_2[z] \rightarrow c_2$ به دست می‌آید. این فرم تعریف تراکنش همه جنبه‌های قابل‌نمایش از اجرای تراکنش را در مدل‌هایش پوشش نمی‌دهد. به‌عنوان‌مثال، این فرم تعریف تراکنش، مقادیر اولیه اقلام داده‌ی یا مقدار نوشته‌شده توسط Write (مقادیر بازنویسی شده) را توصیف نمی‌کند. به‌علاوه، فقط عملیات پایگاه داده را توصیف می‌کند. عملیات دیگر از قبیل عبارت شرطی و یا انتساب در مجموعه عملیات وجود ندارد. ویژگی‌های اجرا که توسط تراکنش‌ها مدل نمی‌شوند **تفسیرنشده**[۷] نامیده می‌شوند که معنی ارضاء نشده را دارد

۳-۲. سابقه‌ها

وقتی مجموعه از تراکنش‌ها به‌طور هم‌روند اجرا می‌شوند، عملیات آن‌ها ممکن است قاطی شوند. چنین اجرایی را با ساختار سابقه مدل می‌کنیم. یک سابقه ترتیب نسبی اجرای عملیات تراکنش نسبت به یکدیگر را نشان می‌دهد. چون، بعضی از این عملیات ممکن است به‌صورت موازی اجرا شوند، یک سابقه به‌عنوان ترتیب جزئی تعریف می‌شود. اگر تراکنش T_i ترتیب دو عملیات خودش را مشخص کند، این دو عمل باید به همان ترتیب در سابقه شامل T_i ظاهر شوند. به‌علاوه، نیاز داریم که یک سابقه ترتیب تمام عملیات برخورد دار[۲] را مشخص کند. **دو عمل را برخورد دار** می‌گویند:

۱. اگر هر دو عمل بر روی یک‌قلم داده یکسانی کار کنند.
۲. حداقل یکی از اعمال Write باشد.
۳. این دو عملیات به دو تراکنش متمایز تعلق داشته باشند. یعنی، $T_i \# T_j$ است (مانند جدول زیر):

---
[7]. Uninterpreted [2]. conflicting operations

T_i / T_j	عمل $r_j(P)$	عمل $w_j(P)$
عمل $r_i(P)$	بی برخورد	برخورد دار
عمل $w_i(P)$	برخورد دار	برخورد دار

بنابراین، Read(x) با Write(x) و Write(x) با هردوی Read(x) و Write(x) برخورد دارد. در اجرای تراکنش‌ها اگر دو عمل برخورد داشته باشند، ترتیب اجرای آن‌ها مهم است. مقدار x که از عمل Read(x) برمی‌گردد، وابسته به این است که عمل خواندن قبل یا بعد از عمل Write(x) خاصی بوده است یا نه. همچنین، مقدار پایانی x وابسته به این است که کدام‌یک از دو عمل Write(x) بعد از دیگری انجام‌شده است.

رسماً، فرض کنید $T = \{T_1, T_2, \ldots T_n\}$ مجموعه‌ی از تراکنش‌ها باشند. یک **سابقه کامل**[8] H روی یک ترتیب جزئی با رابطه $<_H$ نشان داده می‌شود که:

۱. $H = \cup_{i=1}^{n} T_i$. اجرای که با H ارائه می‌گردد دقیقاً شامل عملیات که توسط T_1، T_2، ...و T_n صادرشده است. یعنی، H شامل تمام تراکنش‌ها است.

۲. $<_H \supseteq \cup_{i=1}^{n} <_i$. یعنی، تعداد رابطه‌ی جزئی در بین تراکنش‌ها کم‌تر یا مساوی روابط ترتیب جزئی در سابقه است. زیرا، نه‌تنها در سابقه باید توالی قرار گرفتن دستورات برخورد دار موجود در تراکنش‌ها را بررسی نماییم، بلکه باید دستورات برخورد دار موجود در بین تراکنش‌های مختلف را نیز موردتوجه قرار دهیم. تعداد ترتیب جزئی در سابقه کامل بیش‌تر می‌شود. بهترین حالت زمانی است که اقلام داده مشترک وجود نداشته باشد که در این صورت تعداد ترتیب‌های جزئی تراکنش و سوابق کامل برابر هستند.

۳. به ازای هر دو عمل برخورد دار $p \in H$ و q، یا $p <_H q$ یا $q <_H p$ است. یعنی، هر دو عمل برخورد دار توسط $<_H$ مشخص‌شده است. به‌عبارت‌دیگر، برای هر دو عمل برخورد دار باید ترتیب آن‌ها مشخص گردد.

در سابقه‌ی کامل، اندیس H را از $<_H$ حذف می‌کنیم. یک سابقه، هر پیشوندی از سابقه کامل است. بنابراین، یک سابقه اجرای غیر کامل ممکن از تراکنش‌ها را ارائه می‌دهد. همان‌طور که در فصل اول بیان گردید، ممکن است در اجرای تراکنش خطای رخ دهد. چنین خطای ممکن است اجرای تراکنش فعال را متأثر کند.

برای توصیف این تعریف‌ها، مثال زیر را ببینید:

مثال ۲-۲. سه تراکنش را به صورت زیر در نظر بگیرید و یک سابقه کامل و یک سابقه پیشوندی از آن سابقه بنویسید.

$T_1 = r_1[x] \rightarrow w_1[x] \rightarrow c_1$
$T_3 = r_3[x] \rightarrow w_3[y] \rightarrow w_3[x] \rightarrow c_3$
$T_4 = r_4[y] \rightarrow w_4[x] \rightarrow w_4[y] \rightarrow w_4[z] \rightarrow c_4$

یک سابقه کامل روی تراکنش‌ها $\{T_1, T_3, T_4\}$ عبارت است از :

---

[8]. Complete History

$$H_1 = \begin{array}{l} r_3[x] \rightarrow w_3[y] \rightarrow w_3[x] \rightarrow c_3 \\ \uparrow \quad\quad \uparrow \\ r_4[y] \rightarrow w_4[x] \rightarrow w_4[y] \rightarrow w_4[z] \rightarrow c_4 \\ \quad\quad \uparrow \\ \quad\quad r_1[x] \rightarrow w_1[x] \rightarrow c_1 \end{array}$$

یک سابقه روی همان سه تراکنش که پیشوندی از سابقه H_1 می‌باشد عبارت است از:

$$H_1' = \begin{array}{l} r_3[x] \rightarrow w_3[y] \\ \uparrow \quad\quad \uparrow \\ r_4[y] \rightarrow w_4[x] \rightarrow w_4[y] \\ \uparrow \\ r_1[x] \rightarrow w_1[x] \rightarrow c_1 \end{array}$$

یعنی، سابقه H_1' پیشوندی از سابقه H است.

در این سابقه، پیکان‌هایی که از طریق تعددی به دست می‌آیند، رسم نشده‌اند. اغلب، سوابق را به ترتیب می‌بینیم (مانند سابقه زیر):

$$r_1[x] \rightarrow r_1[x] \rightarrow w_1[x] \rightarrow c_1 \rightarrow w_3[y] \rightarrow w_3[x] \rightarrow c_3$$

به‌طور نرمال، در زمان نوشتن چنین سوابقی پیکان‌ها را حذف می کنیم (مانند زیر):

$$r_1[x]\ r_1[x]\ w_1[x]\ c_1\ w_3[y]\ w_3[x]\ c_3$$

تراکنش T_1 در سابقه H تثبیت (یا سقط) شده است، اگر $c1 \in H$ (یا $a1 \in H$) باشد. در غیر این صورت(اگرنه تثبیت‌شده و نه سقط شده باشد)، T_1 **تراکنش فعال** در H است. البته، یک سابقه کامل هیچ تراکنش فعالی ندارد. یک سابقه H را در نظر بگیرید، **تصویر تثبیت‌شده** H، که با C(H) نشان داده می‌شود، سابقه‌ای است از H که در آن عملیاتی که مربوط به تراکنش‌های تثبیت نشده باشد، حذف‌شده است.

توجه کنید که C(H) یک سابقه کامل روی مجموعه‌ی از تراکنش‌های تثبیت‌شده در H است. اگر H نشان‌دهنده اجرا در زمان خاصی باشد، C(H) فقط بخشی از اجرا می‌باشد که می‌توانیم روی آن حساب بازکنیم، چون تراکنش‌های فعال ممکن است در اثر رخ دادن خطای سیستمی سقط شوند.

۸–۲. مسائل حل‌شده

مثال ۱.کدام‌یک از زمان‌بندی‌های زیر پی‌درپی پذیر برخوردی هستند؟

S_1: $R_1(x)$, $R_3(x)$, $W_1(X)$, $R_2(x)$, $W_3(x)$
S_2: $R_1(x)$, $R_3(x)$, $W_3(X)$, $W_1(x)$, $R_2(x)$
S_3: $R_3(x)$, $R_2(x)$, $W_3(X)$, $R_1(x)$, $W_1(x)$

S1، پی‌درپی پذیر برخوردی نیست. چون R3(X) و W1(X) و همچنین R2(X) و W3(X) را نمی‌توان جابه‌جا کرد (چون برخورد دارند). S2، پی‌درپی پذیر برخوردی نیست. دستورات R1(X) و W3(X)، W1(X) با R3(X)، R1(X) با W3(X) برخورد دارند و نمی‌توان آن‌ها را جابه‌جا کرد. S3، پی‌درپی پذیر برخورد است. چون با جابه‌جایی R3(X) و R2(X) (دو دستور اول) به اجرای پی‌درپی T1 → T3 → T2 می‌رسیم.

مثال ۲. سابقه زیر مشکل تغییر گم‌شده دارد، آیا پی‌درپی پذیر دیدی است؟

$$H = r_1(x)\ r_2(x)\ w_2(x)\ w_1(x)$$

ابتدا مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه H را حساب می کنیم:

$RF(H) = \{\}$
$FW(H) = \{w_1(x)\}$

سپس مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_1 < T_2$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_1 < T_2 = r_1(x)\ w_1(x)\ r_2(x)\ w_2(x)$
$RF(T_1 < T_2) = \{(r_2(x), w_1(x))\}$
$FW(T_1 < T_2) = \{w_2(x)\}$

اکنون مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_2 < T_1$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_2 < T_1 = r_2(x)\ w_2(x)\ r_1(x)\ w_1(x)$
$RF(T_2 < T_1) = \{(r_1(x), w_2(x))\}\}$
$FW(T_2 < T_1) = \{w_1(x)\}$

پس این سابقه پی‌درپی پذیر دیدی نیست. چون RF(H) مخالف هر یک از $RF(T_1 < T_2)$ و $RF(T_2 < T_1)$ است.

مثال ۳. سابقه زیر مشکل خواندن تکرار نشدنی دارد، آیا پی‌درپی پذیر دیدی است؟

$H = r^1_1(x)\ r2(x)\ w2(x)\ r^2_1(x).$

ابتدا مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه H را حساب می‌کنیم:

$RF(H) = \{(r^2_1(x), w_2(x))\}$
$FW(H) = \{w_2(x)\}$

سپس مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_1 < T_2$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_1 < T_2 = r^1_1(x)\ r^2_1(x)\ r_2(x)\ w_2(x)$
$RF(T_1 < T_2) = \{\}$
$FW(T_1 < T_2) = \{w_2(x)\}$

اکنون مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_2 < T_1$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_2 < T_1 = r_2(x)\ w_2(x)\ r^1_1(x)\ r^2_1(x)$
$RF(T_2 < T_1) = \{(r^1_1(x), w_2(x)), (r^2_1(x), w_2(x))\}$
$FW(T_2 < T_1) = \{w_2(x)\}$

این سابقه پی‌درپی پذیر دیدی نیست، چون RF(H) مخالف هر یک از $RF(T_1 < T_2)$ و $RF(T_2 < T_1)$ است.

مثال ٤. سابقه زیر مشکل خواندن داده تثبیت نشده دارد، آیا پی‌درپی پذیر دیدی است؟

$H = r_1(y)\ r_2(y)\ r_2(z)\ w_2(y)\ w_2(z)\ r_1(z).$

ابتدا مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه H را حساب می‌کنیم:

$RF(H) = \{(r_1(z), w_2(z))\}$

$FW(H) = \{w_2(y), w_2(z)\}$

سپس مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_1 < T_2$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_1 < T_2 = r_1(y)\ r_1(z)\ r_2(y)\ r_2(z)\ w_2(y)\ w_2(z)$

$RF(T_1 < T_2) = \{\}$

$FW(T_1 < T_2) = \{w_2(y), w_2(z)\}$

اکنون مجموعه خواندن از (RF) و مجموعه نوشتن نهایی سابقه $T_1 < T_2$ را حساب می‌کنیم(یعنی، سابقه زیر):

$T_2 < T_1 = r_2(y)\ r_2(z)\ w_2(y)\ w_2(z)\ r_1(y)\ r_1(z)$

$RF(T_2 < T_1) = \{(r_1(y), w_2(y)), (r_1(z), w_2(z))\}$

$FW(T_2 < T_1) = \{w_2(y), w_2(z)\}$

این سابقه پی‌درپی پذیر دیدی نیست، چون RF(H) مخالف هر یک از $RF(T_1 < T_2)$ و $RF(T_2 < T_1)$ است.

مثال ۵. کدام‌یک از زمان‌بندی‌های زیر پی‌درپی پذیر دیدی (VSR) هستند؟

(1) : $r_1(x), r_2(y), w_1(y), r_2(x), w_2(x)$
(2) : $r_1(x), r_2(y), w_1(x), w_1(y), r_2(x), w_2(x)$
(3) : $r_1(x), r_1(y), r_2(y), w_2(z), w_1(z), w_3(z), w_3(x)$
(4) : $r_1(y), r_1(y), w_2(z), w_1(z), w_3(z), w_3(x), w_1(x)$

گزینه ۱، یعنی $(r_1(x), r_2(y), w_1(y), r_2(x), w_2(x))$، VSR نیست. زیرا، دو زمان‌بندی پی‌درپی زیر را در نظر بگیرید:

S_1 : $r_1(x), w_1(y), r_2(y), r_2(x), w_2(x)$
S_2 : $r_2(y), r_2(x), w_2(x), r_1(x), w_1(y)$

این دو زمان‌بندی پی‌درپی، هم‌ارز دیدی نیستند، آن‌ها رابطه **خواندن – از** (READ- FROM) متفاوت دارند.

گزینه (۲)، یعنی $(r_1(x), r_2(y), w_1(x), w_1(y), r_2(x), w_2(x))$ VSR نیست، زیرا، زمان‌بندهای زیر را مشاهده کنید:

S_1: $r_1(x), w_1(x), w_1(y), r_2(y), r_2(x), w_2(x)$

S_1: $r_2(y), r_2(x), w_2(x), r_1(x), w_1(x), w_1(y)$

این دو زمان‌بندی رابطه **خواندن – از** متفاوت دارند.

گزینه 3، یعنی، $(r_1(x), r_1(y), r_2(y), w_2(z), w_1(z), w_3(z), w_3(x))$، VSR است، زیرا، هم‌ارز دیدی با زمان‌بندی پی‌درپی زیر است:

S : $r_2(y), w_2(z), r_1(x), r_1(y), w_1(z), w_3(z), w_3(x)$

گزینه ۴، یعنی $(r_1(y), r_1(y), w_2(z), w_1(z), w_3(z), w_3(x), w_1(x))$، VSR نیست، زیرا، در آن یک زمان‌بندی پی‌درپی نمی‌تواند رابطه نوشتن نهایی (FINAL WRITE) یکسان موجود باشد. توجه کنید که تراکنش یک نوشتن نهایی روی X و همچنین یک نوشتن روی Z دارد، درحالی‌که تراکنش ۳، یک نوشتن نهایی روی Z و همچنین یک نوشتن روی X دارد.

مثال ٦. تعیین کنید زمان‌بندی‌های زیر در کدام‌یک از کلاس‌های Non-VSR، VSR، CSR قرار دارند. در هر حالتی از زمان‌بندی که هردو VSR و CSR باشد، تمام زمان‌بندی‌های پی‌درپی هم‌ارز با آن‌ها را پیدا کنید .

(1) : $r_1(x), w_1(x), r_2(z), r_1(y), w_1(y), r_2(x), w_2(x), w_2(z)$
(2) : $r_1(x), w_1(x), w_3(x), r_2(y), r_3(y), w_3(y), w_1(y), r_2(x)$
(3) : $r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), w_3(y), w_3(x), w_1(y), w_5(x), w_1(z), w_5(y), r_5(z)$
(4) : $r_1(x), r_3(y), w_1(y), w_4(x), w_1(t), w_5(x), r_2(z), r_3(z), w_2(z), w_5(z), r_4(t), r_5(t)$
(5) : $r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), r_3(y), r_3(x), w_1(y), w_5(x), w_1(z), r_5(y), r_5(z)$
(6) : $r_1(x), r_1(t), r_3(z), r_4(z), w_2(z), r_4(x), r_3(x), w_4(x), w_4(y), w_3(y), w_1(y), w_2(t)$
(7) : $r_1(x), r_4(x), w_4(x), r_1(y), r_4(z), w_4(z), w_3(y), w_3(z), w_1(t), w_2(z), w_2(t)$

گزینه (۱)، هر دو VSR و CSR است، و هم‌ارز برخوردی آن به‌صورت زیر است:

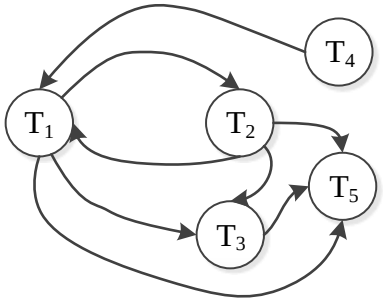
S : $r_1(x), w_1(x), r_1(y), w_1(y), r_2(z), r_2(x), w_2(x), w_2(z)\ r1(x), w_1(x), w_3(x), r_2(y), r_3(y), w_3(y), w_1(y), r_2(x)$

زمان‌بندی گزینه (۲)، یک زمان‌بندی Not-VSR (غیر VSR) است.

در زمان‌بندی گزینه (۳)، یعنی:

$r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), w_3(y), w_3(x), w_1(y), w_5(x), w_1(z), w_5(y), r_5(z)$

گراف اولویت آن به شکل زیر است:

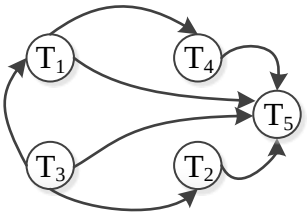


این زمان‌بندی CSR نیست، چون در گراف برخورد آن دور وجود دارد. این زمان‌بندی همچنین VSR نیست، زیرا، برای داده X، در تراکنش T_1 باید قبل از تراکنش T_2 بی آید و تراکنش T_2 باید قبل از تراکنش T_1 بی‌آید (این دو تراکنش هر دو قبل از هر عمل نوشتن مقدار X را می‌خوانند).

گراف برخورد، گزینه (۴)، یعنی:

$r_1(x), r_3(y), w_1(y), w_4(x), w_1(t), w_5(x), r_2(z), r_3(z), w_2(z), w_5(z), r_4(t), r_5(t)$

به شکل زیر است:



این زمان‌بندی هر دو CSR و VSR است. زمان‌بندی‌های پی‌درپی هم‌ارز آن عبارت‌اند از:

S_1 : $r_3(y), r_3(z), r_1(x), w_1(y), w_1(t), r_2(z), w_2(z), w_4(x), r_4(t), w_5(x), w_5(z), r_5(t)$
S_2 : $r_3(y), r_3(z), r_2(z), w_2(z), r_1(x), w_1(y), w_1(t), w_4(x), r_4(t), w_5(x), w_5(z), r_5(t)$

گزینه (۵)، یعنی:

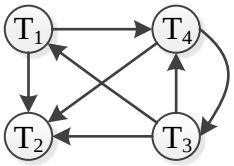
$r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), r_3(y), r_3(x), w_1(y), w_5(x), w_1(z), r_5(y), r_5(z)$

یک زمان‌بندی VSR نیست. زیرا، دو تراکنش T_1 و T_2 با هم X را می‌خوانند و می‌نویسند، اما در این زمان‌بندی قبل از هر عملیات نوشتن، X را می‌خوانند، و بنابراین هر زمان‌بندی پی‌درپی با تراکنش‌های T_1 و T_2 رابطه **خواندن از** متفاوت را خواهند داشت.

گراف برخورد گزینه (۶)، یعنی:

$r_1(x), r_1(t), r_3(z), r_4(z), w_2(z), r_4(x), r_3(x), w_4(x), w_4(y), w_3(y), w_1(y), w_2(t)$

به شکل زیر است:

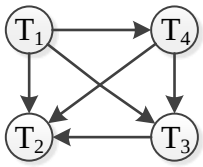


این زمان‌بندی CSR نیست. این زمان‌بندی VSR نیز نیست (تراکنش T_1، روی‌داده Y، نوشتن نهایی دارد، و بنابراین تراکنش T_1 باید بعد از تراکنش T_4 بی‌آید. اما، تراکنش T_4 روی‌داده X می‌نویسد، پس باید بعد از تراکنش T_1 قرار گیرد).

گراف برخورد گزینه (۷)، یعنی:

$r_1(x), r_4(x), w_4(x), r_1(y), r_4(z), w_4(z), w_3(y), w_3(z), w_1(t), w_2(z), w_2(t)$

به شکل زیر است:



این زمان‌بندی هردوی CSR و VSR است. هم‌ارز زمان‌بندی پی‌درپی عبارت است از:

S : $r_1(x), r_1(y), w_1(t), r_4(x), w_4(x), r_4(z), w_4(z), w_3(y), w_3(z), w_2(z), w_2(t)$

۹-۲. سؤالات چهارگزینه‌ای

۱. توالی پذیری تراکنش‌ها چه مزیتی دارد؟

الف: صحت اجرای هم‌زمان تراکنش‌ها
ب: سرعت تراکنش‌های فقط خواندنی
ج: پایداری (Durability) تراکنش‌ها
د: اجرای تراکنش‌ها به ترتیب زمان شروع آن‌ها

۲. ضعیف توالی پذیری با کدام‌یک از اهداف زیر صورت می‌گیرد؟

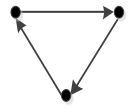
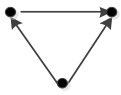
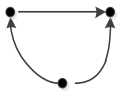
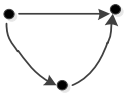
الف: جلوگیری از بن‌بست
ب: جلوگیری از قحطی
ج: افزایش هم‌زمانی
د: جبران‌پذیری(ترمیم‌پذیری) تراکنش‌ها

۳. اگر VSR مجموعه طرح‌های توالی پذیر نمایی و CSR مجموعه طرح‌های توالی پذیر تعارضی باشد، کدام مورد درست است؟

الف : $CSR \subset VSR$
ب: $CSR \subseteq VSR$
ج: $VSR \subset CSR$
د: $VSR \subseteq CSR$

٤. با توجه به طرح S گراف پیشایندی کدام است؟

$R_3(Y), R_3(Z), R_1(X), W_1(X), W_3(Y), W_3(Z), R_2(Z), R_1(Y), W_2(X), R_2(Y), W_2(Y), R_2(X), W_2(X);$

الف: 
ب: 
ج: 
د: 

۵. تراکنش T را می‌توان با یک ترتیب جزئی به‌صورت $T = \{\sum, \varepsilon\}$ نمایش داد. برای تراکنش زیر، مجموعه ε چند عضو دارد؟

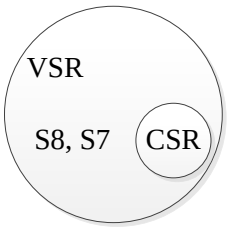
```
BEGIN TRANSACTION
R(D1)
R (D2)
D:= D1 + D2
W(D1)
COMMIT
```

الف: ۷
ب: ۶
ج: ۵
د: ۴

۱۰-۲. پاسخ تشریحی سؤالات چهارگزینه‌ای

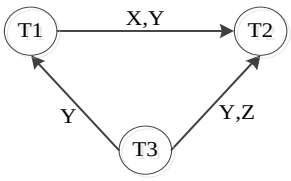
۱. **گزینه (الف) صحیح است.** چون، معروف‌ترین روش کنترل هم‌روندی (صحت اجرای هم‌زمان تراکنش‌ها) پی‌درپی پذیری (توالی پذیری) است.

۲. **گزینه (ج) صحیح است.**



۳. **گزینه (الف) صحیح است.** سریالی پذیر برخوردی (CSR)، اگر دستورات خواندن و نوشتن مربوط به تراکنش‌های مختلف به‌صورت هم‌روند اجرا شوند، بعضاً باهم برخورد دارند و بعضاً ندارند. اما سریال پذیری دیدی (VSR)، زمان‌بند s سریالی پذیر دیدی است، اگر معادل دیدی با یک زمان‌بند سریالی باشد. محدوده عملکرد دو روش اصلی سریالی پذیر ی شکل روبه‌رو است:

٤. گزینه (ب) صحیح است.



$(R_2(Z),W_3(Z)),\quad (R_2(Y),W_3(Y)),\quad (R_2(Y),R_1(1)$
$(W_2(Y),R_3(Y)),(W_2(Y),W_3(Y),W_2(Y),R_2(Y)),$
$(W_1(Y),W_1(X)(R_3(Y),W_1(Y),R_1(X),W_2(X)),(W_1(X),R_2(X)),$
$(W_1(X),W_2(X)$
$(W_3(Y),R_1(Y)),\quad (W_3(Y),W_1(Y)) \Rightarrow T_3 \rightarrow T_1 \rightarrow T_2$

٥. گزینه (ج) صحیح است. زیرا:

$$\sum = \{R(D1), R(D2), W(D1), c\}$$

$$\varepsilon = \{(R(D1), W(D1)), (R(D2), W(D1)), (W(D1), C), (R(D1), C), (R(D2), C)\}$$

فصل ۳ قفل‌گذاری دومرحله‌ای

۱-۳. زمان‌بندی‌های تهاجمی و محافظه‌کارانه

در این فصل، مشهورترین زمان‌بند محصولات تجاری، همان زمان‌بندهای **قفل‌گذاری دومرحله‌ای** (2PL) را بررسی می‌کنیم. در اکثر بخش‌های این فصل، روی قفل‌گذاری سیستم پایگاه داده متمرکز که مدل آن را در فصل اول مشاهده کردید، تمرکز می‌کنیم. در بخش پایانی، پروتکل‌های قفل‌گذاری خاصی را برای درخت‌ها بحث می‌نمایم. در فصل اول بیان گردید که وقتی یک زمان‌بند عملیاتی را از مدیر تراکنش خودش (TM) دریافت می‌کند، یکی از سه امکان زیر وجود دارد:

۱. بلافاصله آن را زمان‌بندی می‌کند (آن را به مدیر داده (DM) می‌فرستد).
۲. آن را به تعویق (تأخیر) می‌اندازد (آن را در صف قرار می‌دهد).
۳. آن را رد می‌کند (بدین‌وسیله موجب سقط‌شدن تراکنش می‌گردد).

هر نوع زمان‌بند معمولاً به یک یا دو امکان از سه امکان بالا توجه می‌کند. بر اساس امکان‌هایی که زمان‌بند توجه می‌کند، می‌توان یک تمایز فازی (نادقیق) بین زمان‌بندهای **مهاجم**[۹] و **محافظه‌کارانه**[۲] برقرار کرد.

یک زمان‌بند مهاجم، تمایل دارد تا از تعویق عملیات اجتناب کند. یعنی، سعی می‌کند آن‌ها را فوراً زمان‌بندی نماید، اما برای انجام بقیه عملیات به این فرم، احتمالاتی جهت ذخیره عملیاتی که بعداً دریافت می‌نماید را مقدم می‌کند. با حذف احتمال عملیات ذخیره‌شده ممکن است در شرایطی گیر کند که در آن امیدی به اتمام اجرای تمام تراکنش‌های فعال به‌صورت پی‌درپی داشته باشد. در این نقطه، باید جهت عملیات رد کردن یک یا چند تراکنش عمل مرتب‌سازی را انجام دهد، بنابراین سبب سقط آن‌ها می‌شود.

از طرف دیگر زمان‌بند محافظه‌کارانه، تمایل زیادی به تأخیر عملیات دارد. پس در تغییر ترتیب عملیات که بعداً دریافت می‌کند، آزادی بیش‌تری دارد.

---

[9]. Aggressive [2].Conservative

حالت افراطی یک زمان‌بند محافظه‌کارانه این است که عملیات همه تراکنش‌ها به‌غیراز یک تراکنش را در هرلحظه به تعویق بی اندازد. وقتی اجرای این تراکنش خاتمه می‌یابد، تراکنش دیگری انتخاب می‌شود تا عملیاتش پردازش گردد. چنین زمان‌بندی تراکنش‌ها را به‌صورت پی‌درپی پردازش می‌کند. این زمان‌بند، هرگز نیازی به رد یک عمل ندارد، اما گاهی اوقات از چنین ردهای به‌وسیله تعویق خیلی زیاد (بیش‌ازحد) عملیات جلوگیری می‌کند.

بین دو زمان‌بند مهاجم و محافظه‌کار ازلحاظ کارایی مصالحه‌ای وجود دارد. زمان‌بندهای مهاجم از تعویق عملیات اجتناب می‌کنند و درنتیجه ریسک (احتمال) سقط عمل بعدی آن‌ها وجود دارد. زمان‌بندی‌های محافظه‌کار عمداً با تعویق عملیات از سقط آن‌ها اجتناب می‌کنند.

هر یک از این روش‌های زمان‌بندی برای نوع خاصی از برنامه‌های کاربردی خوب کار می‌کنند. به‌عنوان‌مثال، در برنامه کاربردی که تراکنش‌های هم‌روند به‌ندرت برخورد دارند، یک زمان‌بند مهاجم ممکن است بهتر از زمان‌بند محافظه‌کار اجرا شود. چون در این نوع برنامه‌های کاربردی برخوردها نادر هستند، پس برخوردهایی که نیاز به رد یک عملیات دارند نیز نادرتر می‌باشند. بنابراین، زمان‌بند مهاجم، اکثر اوقات عملیات را رد نمی‌کند. در مقابل، یک زمان‌بند محافظه‌کار بدون نیاز به رد، عملیات را به تأخیر می‌اندازد، برخوردهایی که به‌ندرت اتفاق می‌افتند را پیش‌بینی می‌کند.

از طرف دیگر، در برنامه کاربردی که تراکنش‌های هم‌روند برخورد دارند، تقریباً همه انواع زمان‌بندها، نسخه (گونه) محافظه‌کار و مهاجم دارند. کلاً زمان‌بند محافظه‌کار باید رفتار آینده تراکنش‌ها را به‌منظور آمادگی برای عملیاتی که هنوز نرسیده‌اند، پیش‌بینی کند. اطلاعاتی اصلی که نیاز دارد بداند، مجموعه اقلام داده‌ای است که هر تراکنش می‌خواند یا می‌نویسد (به ترتیب مجموعه **خواندن از** و مجموعه **نوشتن به** تراکنش نامیده می‌شوند).

از این طریق، زمان‌بند می‌تواند پیش‌بینی کند که کدام‌یک از عملیات در حال زمان‌بندی ممکن است با عملیاتی که در ادامه می‌رسند، برخورد داشته باشند. در مقابل، از آنجائی که زمان‌بند مهاجم با رسیدن عملیات آن‌ها را فوراً زمان‌بندی می‌کند و برای تصحیح اشتباهات روی رد عملیات تأکید دارد، به چنین اطلاعاتی نیاز ندارد.

اگر تراکنش‌ها از قبل مجموعه‌های خواندن از و مجموعه‌های نوشتن‌شان به را اعلام کنند، معمولاً یک نسخه بسیار محافظه‌کار از هر نوع زمان‌بند می‌تواند ساخته شود. یعنی، مدیر تراکنش (TM) در هنگام شروع پردازش تراکنش، مجموعه خواندن از و مجموعه نوشتن به را به زمان‌بند می‌دهد.

یک اشکالی که برای ساخت زمان‌بندهای بسیار محافظه‌کار وجود دارد این است که اجراهای متفاوت برنامه ممکن است به مجموعه‌های از اقلام داده متفاوت دستیابی داشته باشند و نتایج مختلفی ارائه دهند. این‌زمانی اتفاق می‌افتد که در برنامه‌ها عبارت شرطی وجود داشته باشد. برای مثال، برنامه زیر بر اساس مقداری که برای x وارد می‌شود یا x و y یا x و z را می‌خواند.

```
procedure fuzzy_readset
 begin
    start;
    a:= read (x);
```

```
    if (a > 0) then b:= read(y) else    b:= read(z);
    commit
end
```

در این حالت، تراکنش باید مجموعه تمام اقلام داده‌ای را که ممکن است بخواند یا بنویسد، قبلاً اعلام کند. اغلب این عمل موجب می‌شود تا تراکنش درباره مجموعه خواندن از و مجموعه نوشتن به غلو کند. به‌عنوان‌مثال، در اجرای تراکنش fuzzy_readset، حتی برای هر اجرای تنها اگر به دو قلم داده دسترسی داشته باشند، باید مجموعه خواندن از را به‌صورت {x, y, z} اعلام نماید. این همان مشکلی است که وقتی تراکنش-ها با سیستم پایگاه داده با استفاده از زبان پرس‌وجوی (به‌عنوان‌مثال، رابطه‌ای) سطح بالا تعامل می‌کنند، ممکن است، رخ دهد. یک پرس و جوی سطح بالا بالقوه به بخش‌های بزرگی از پایگاه داده دسترسی دارد، حتی هرچند در هر اجرا پرس و جوی سطح بالا فقط به بخش کوچکی از پایگاه داده دسترسی دارد.

زمان‌بندی محافظه‌کار درصورت نیاز، اجرای دستورات تراکنش را به تعویق می‌اندازد و محتاطانه و محافظه‌کارانه اجرای دستورات را پی می‌گیرد تا حتی‌الامکان بتواند دستورات را اجرا و از سقط آن‌ها جلوگیری کند.
زمان‌بندی محافظه‌کار تأخیر دارد، اما حتی امکان سقط و رد کردن تراکنش ندارد. بالعکس، در زمان‌بندی مهاجم، هدف پرهیز از تأخیر در اجرای تراکنش‌ها است، لذا، دستورات را فوراً اجرا می‌کند که البته ممکن است مشکلاتی پیش بیاید و مجبور به سقط برخی از تراکنش‌ها گردد. به‌طورکلی، زمان‌بندی‌های محافظه‌کار که از سقط تراکنش جلوگیری می‌کنند، عملکرد بهتری دارند. به‌هرحال، الگوریتم‌ها و پروتکل‌های کنترل هم‌روندی را می‌توان در قالب یکی از این دو رویکرد پیاده‌سازی نمود.

۲-۳. قفل‌گذاری دومرحله‌ای پایه

قفل مکانیسمی است که معمولاً جهت حل مشکل دستیابی هم‌زمان به داده مشترک استفاده می‌شود. ایده اصلی قفل‌گذاری ساده است. به هر داده یک قفل تخصیص می‌یابد. قبل از این‌که تراکنش T_1 به داده‌ای دسترسی یابد، زمان‌بند ابتدا قفل تخصیص‌یافته به آن را بررسی می‌کند. اگر تراکنش هیچ قفلی نداشته باشد، زمان‌بند، آن را برای تراکنش T_1 قفل می‌کند. اگر تراکنش دیگری نظیر T_2 وجود داشته باشد که داده را قفل کرده باشد، آن گاه تراکنش T_1 باید تا زمان آزاد کردن قفل توسط T_2 منتظر بماند. یعنی، زمان‌بند تا زمانی که T_2 قفل را آزاد نکند، قفل را به T_1 نخواهد داد. درنتیجه، زمان‌بند مطمئن می‌شود که در هرلحظه فقط یک تراکنش می‌تواند روی‌داده قفل داشته باشد، بنابراین، در هرلحظه فقط یک تراکنش می‌تواند به قلم داده دستیابی داشته باشد. قفل‌گذاری می‌تواند برای اطمینان پی‌درپی پذیری به‌وسیله زمان‌بند استفاده شود.

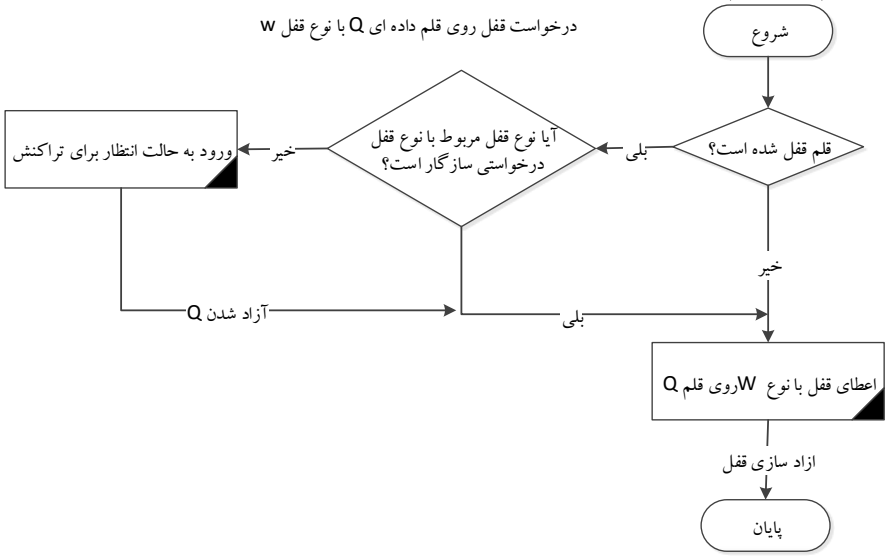
به‌طورکلی دو نوع قفل زیر مرسوم است:

۱. **قفل دوحالتی باینری**[10]: در این نوع قفل، سازگاری بین قفل‌ها وجود ندارد؛ یعنی اگر داده‌ای توسط تراکنشی قفل‌شده باشد، به‌هیچ‌وجه تراکنش دیگری نمی‌تواند آن را قفل کند و اگر قفل نشده باشد می تواند آن را قفل کند.

[10]. binary lock

۲. **قفل اشتراکی – انحصاری :** در این نوع قفل، قفل‌ها به دو نوع اشتراکی(S) و انحصاری(X) تقسیم می‌شوند. قفل اشتراکی مخصوص خواندن و قفل انحصاری برای نوشتن است. قفل اشتراکی قابل به اشتراک گذاری بین چندین تراکنش می‌باشد. یعنی، چند تراکنش می‌توانند هم‌زمان یک داده را بخوانند.

جهت ارائه پروتکل قفل‌گذاری، به برخی از نمادها نیاز داریم. تراکنش‌ها برای خواندن یا نوشتن به اقلام داده‌ی دسترسی می‌یابند. درنتیجه، دو نوع قفل برای اقلام داده‌ی تخصیص می‌یابد: قفل خواندن و قفل نوشتن. از $rl_i[x]$ (یک قفل خواندن روی‌داده x که برخی از کتاب‌ها این قفل اشتراکی می‌نامند و با $S_i[x]$ نشان می‌دهند) و $wl_i[x]$ (یک قفل نوشتن روی‌داده x که برخی از کتاب‌ها این قفل انحصاری می‌نامند و با $X_i[x]$ نشان می‌دهند) استفاده می‌کنیم. از $r1_i[x]$ (یا $wl_i[x]$) برای تعیین تراکنش T_i که قفل خواندن (یا نوشتن) را روی x کسب کرده است، استفاده می‌کنیم. در فصل دوم، از حروف o، p و q برای معرفی هر نوع عملیات (read(r) یا write(x)) استفاده کردیم. از $ol_i[x]$ برای معرفی یک قفل از نوع o به‌وسیله تراکنش T_i روی x استفاده می‌کنیم. الگوریتم اخذ قفل نوشتن در شکل ۱-۳ آمده است.



شکل ۱-۳ الگوریتم درخواست قفل نوشتن.

قفل‌ها می‌توانند به‌عنوان درایه‌های در یک جدول قفل در نظر گرفته شوند. به‌عنوان‌مثال، $rl_i[x]$ متناظر با درایه $[x, r, T_i]$ در جدول قفل است. به‌هرحال، بیان جزییات ساختار داده در جدول مهم نیست. این جزئیات را در ادامه بحث خواهیم کرد.

دو قفل $pl_i[x]$ و $ql_i[y]$ برخورد دارند، اگر $x = y$، $i \neq j$ باشد، و عملیات p و q از نوع برخورد دار باشند. یعنی، دو قفل وقتی برخورد دارند که روی‌داده یکسانی عمل کنند، توسط تراکنش‌های متفاوتی صادرشده باشند و یکی از آن دو قفل نوشتن باشد. بنابراین، دو قفل روی‌داده‌های مختلف یا روی‌داده یکسان که توسط یک تراکنش صادرشده باشد، حتی اگر (هرچند) آن دو قفل برخورد دار باشند، برخورد ندارند.

	S یا (rl)	X یا (wl)

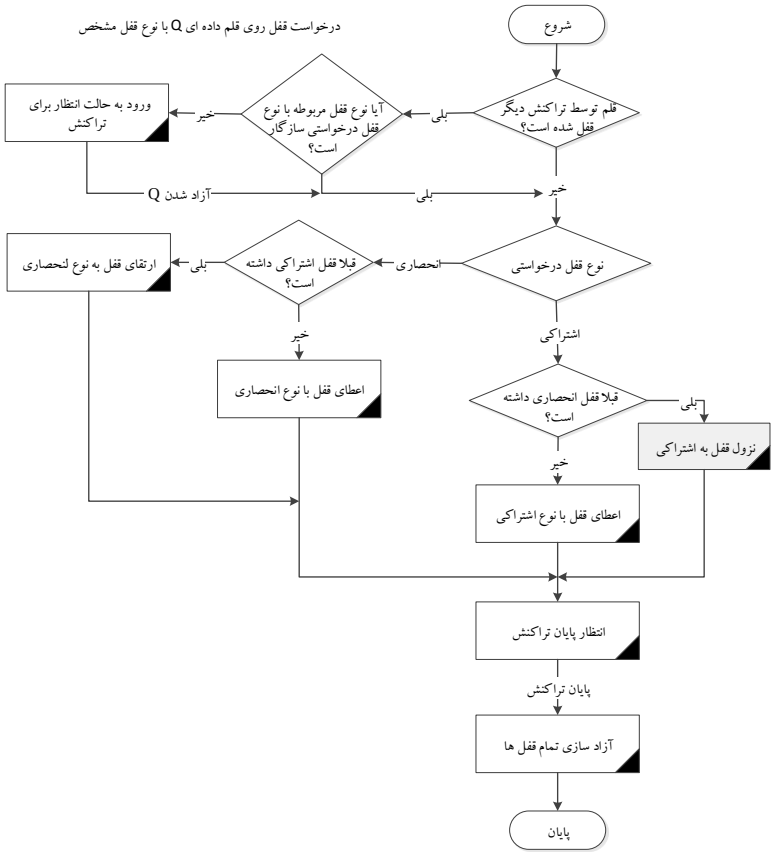
با برخورد	بی برخورد	S یا (rl)
با برخورد	با برخورد	X یا (wl)

وظیفه زمان‌بند 2PL، مدیریت اخذ و آزادسازی قفل‌ها توسط تراکنش‌ها است. در این بخش روی نسخه 2PL پایه‌ای تمرکز کرده و در ادامه به نسخه‌های خاص آن می‌پردازیم.

قوانین زیر برای مدیریت و استفاده از قفل‌ها در زمان‌بند 2PL به کار می‌روند(شکل ۲-۳ الگوریتم قفل‌گذاری را نشان می‌دهد):

۱. وقتی زمان‌بند عمل $p_i[x]$ را از مدیر تراکنش (TM) دریافت می‌کند، بررسی می‌کند که آیا $pl_i[x]$ با هر یک از $ql_i[x]$ که قبلاً اخذشده‌اند برخورد دارد یا نه؟ اگر برخورد داشته باشد $p_i[x]$ را به تعویق می‌اندازد و T_i را مجبور می‌کند تا آزادسازی قفل مربوطه منتظر بماند. وگرنه، قفل $pl_i[x]$ اخذ می‌گردد و $p_i[x]$ به مدیر داده (DM) ارسال می‌گردد. این قانون، از دسترسی برخورد دار دو تراکنش هم‌روند جلوگیری می‌کند. بنابراین، اعمال برخورد دار به همان شکلی زمان‌بندی می‌گردند که قفل‌های مربوط به آن‌ها اخذ می‌شوند.

۲. به‌محض این‌که زمان‌بند قفلی را برای T_i اخذ کرد (مانند $pl_i[x]$)، ممکن نیست قفل را حداقل تا زمانی که مدیر داده (DM) پردازش عمل متناظر با $p_i[x]$ را تأیید نکند، آزاد نماید. این **قانون**، را بر مدیریت تحمیل می‌کند که اطمینان دهد تا مدیر داده (DM) اعمال را به همان ترتیبی که زمان‌بند ارسال می‌کند، پردازش نماید. به‌عنوان‌مثال، فرض کنید T_i قفل $rl_i[x]$ را اخذ می‌کند و قبل از این‌که مدیر داده (DM) پردازش $r_i[x]$ را تأیید نماید، آن را آزاد کند (یعنی، قفل $rl_i[x]$ را آزاد کند). بنابراین، T_i امکان اخذ قفل برخورد دار روی x (یعنی $wl_j[x]$) را دارد و $w_j[x]$ را به مدیر داده (DM) می‌فرستد. هرچند زمان‌بند $w_j[x]$ را بعد از $r_i[x]$ به مدیر داده (DM) فرستاده است. بدون قانون (۲) هیچ تضمینی وجود ندارد که مدیر داده اعمال را به همان ترتیبی که دریافت کرد، پردازش نماید.

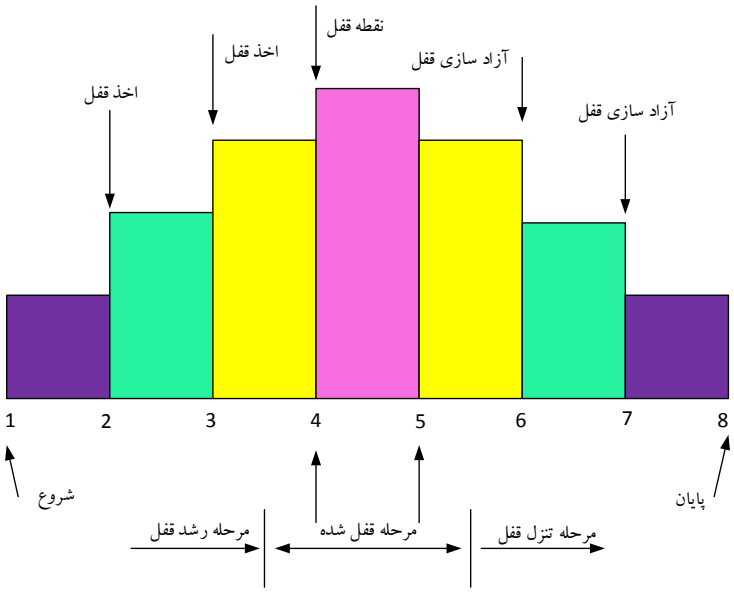


شکل ۲-۳ الگوریتم کامل قفل‌گذاری.

۳. به‌محض این‌که زمان‌بند قفلی را برای یک تراکنش آزاد کرد، بعدازآن نمی‌تواند برای آن تراکنش (روی هیچ قلم داده) قفلی را اخذ نماید. این قانون، **قانون دومرحله‌ای** نامیده می‌شود. این قانون می‌گوید که هر تراکنش ممکن است به دو مرحله تقسیم شود که عبارت‌اند از(شکل ۳-۳):

مرحله رشد[11]، که در آن قفل‌ها اخذ می‌گردند و **مرحله آزادسازی یا تنزل**[2]، که در مدت آن قفل‌های اخذشده آزاد می‌گردند.

[11]. Growing [2]. shrinking



شکل ۳-۳ نمایش مراحل اخذ قفل دو مرحله‌ای.

وظیفه قانون (۳) تضمین این است که همه اعمال برخورد دار از دو تراکنش با ترتیب یکسانی مرتب شوند. مثال زیر را ببینید.

مثال ۱-۳. دو تراکنش T_1 و T_2 و سابقه H_1 را به صورت زیر در نظر بگیرید و در مورد پی در پی پذیری و قوانین قفل گذاری دو مرحله‌ای آن بحث کنید.

T_1: $r_1[x] \rightarrow w_1[y] \rightarrow c_1$
T_2: $w_2[x] \rightarrow w_2[y] \rightarrow c_2$
H_1=$rl_1[x]\ r_1[x]\ ru_1[x]\ wl_2[x]\ w_2[x]\ wu_2[x]\ wl_2[y]\ w_2[y]\ wu_2[y]\ c_2\ wl_1[y]\ w_1[y]\ wu_1[y]\ c1$

چون $r_1[x] < w_2[x]$ و $w_2[y] < w_1[y]$ است، $SG(H_1)$ شامل حلقه $T_1 \rightarrow T_2 \rightarrow T_1$ است و لذا، پی‌درپی پذیر نیست.

مشکلی که در H_1 وجود دارد این است که T_1 قفلی را آزادکرده است ($ru_1[x]$) و بعد آن‌یک قفل دیگر را اخذ کرده است ($wl_1(y)$)، این مسئله با قانون (۳) تناقض دارد.

بین $ru_1[x]$ و $wl_1[y]$ تراکنش دیگر T_2 هم‌روی x و هم‌روی y نوشته است و بنابراین ازلحاظ x باید T_2 بعد از T_1 و ازلحاظ y، T_2 باید قبل از T_1 بی‌آید. اگر T_1 از قانون ۳ پیروی می‌کرد، اعمال بین $ru_1[x]$ و $wl_2[y]$ نباید وجود داشته باشد و بنابراین T_1 نمی‌توانست آنچه را که در H_1 وجود دارد، انجام دهد. یعنی، تراکنش‌های T_1 و T_2 می‌توانستند به شکل زیر اجرا شوند:

۱. در شروع، هیچ قفلی را تخصیص نداده‌اند.

۲. زمان‌بند $r_1[x]$ را از مدیر تراکنش (TM) دریافت می‌کند، قفل $rl_1[x]$ را اخذ کرده و $r_1[x]$ را به مدیر داده (DM) می‌فرستد. در ادامه، مدیر داده (DM) پردازش $r_1[x]$ را تأیید می‌کند. پس جدول قفل به‌صورت زیر تغییر می‌یابد.

قلم داده	نوع قفل	شماره تراکنش
X	r_1	T_1

۳. زمان‌بند $w_2[x]$ را از مدیر تراکنش دریافت می‌کند. نمی‌تواند $wl_2[x]$ را اخذ کند، چون با $rl_1[x]$ برخورد دارد. بنابراین، اجرای $w_2[x]$ را با قرار دادن در صف به تعویق می‌اندازد. بنابراین، صف قفل به‌صورت جدول زیر تغییر می‌یابد.

قلم داده	نوع قفل	شماره تراکنش منتظر
X	w_1	T_2

۴. زمان‌بند $w_1[y]$ را از مدیر تراکنش (TM) دریافت می‌کند. $wl_1[y]$ را اخذ کرده، و $w_1[y]$ را به مدیر داده (DM) تحویل می‌دهد. سپس، مدیر داده (DM) پردازش $w_1[y]$ را تصدیق می‌کند. پس، جدول قفل به‌صورت زیر تغییر می‌یابد.

قلم داده	نوع قفل	شماره تراکنش
X	r_1	T_1
Y	w_1	T_1

۵. زمان‌بند C_1 را از مدیر تراکنش دریافت می‌کند و آن را به مدیر داده می‌فرستد. پس از تصدیق پردازش C_1 توسط مدیر داده، زمان‌بند قفل‌های $rl_1[x]$ و $wl_1[y]$ را آزاد می‌کند. این عمل طبق قانون ۲ امکان‌پذیر است. چون دو عمل $r_1[x]$ و $w_1[y]$ قبلاً پردازش‌شده‌اند. طبق قانون ۳، T_1 درخواست قفلی نکرده است. اکنون جدول قفل خالی می‌شود تا زمان‌بند قفل $wl_2[x]$ که به تعویق افتاده بود و در صف قرار دارد را اخذ کند.

۶. زمان‌بند $wl_2[x]$ را اخذ می‌کند. جدول قفل به‌صورت زیر تغییر می‌یابد و صف قفل خالی می‌گردد:

قلم داده	نوع قفل	شماره تراکنش
X	w_1	T_2

$w_2[x]$ با تأخیر به مدیر داده ارسال می‌گردد و مدیر داده اجرا $w_2[x]$ را تأیید می‌کند.

۷. زمان‌بند $w_2[y]$ را از مدیر تراکنش دریافت می‌کند، قفل $wl_2[y]$ را اخذ کرده تا جدول قفل به‌صورت زیر تغییر یابد:

قلم داده	نوع قفل	شماره تراکنش
X	w_1	T_2
Y	w_1	T_2

$w_2[y]$ را به مدیر داده ارسال می‌کند. سپس، مدیر داده (DM)، پردازش $w_2[y]$ را تأیید می‌نماید.

۸. تراکنش T_2 خاتمه می‌یابد و مدیر تراکنش (DM) c_2 را به زمان‌بند می‌فرستد. زمان‌بند c_2 را به مدیر داده می‌فرستد. پس از تصدیق مدیر داده مبتنی بر پردازش c_2 زمان‌بند قفل‌های $wl_2[x]$ و $wl_2[y]$ را آزاد می‌کند. اکنون جدول قفل خالی‌شده و این اجرا در سابقه زیر آمده است:

$H_2= rl_1[x]\ r_1[x]\ wl_1[y]\ w_1[y]\ c_1\ ru_1[x]\ wu_1[y]\ wl_2[x]\ w_2[x]\ wl_2[y]\ w_2[y]\ c_2\ wu_2[x]\ wu_2[y]$

H_2 پی‌درپی است و بنابراین پی‌درپی پذیر (SR) است.

مشکل اصلی زمان‌بندی‌های 2PL بن‌بست(dead lock) است. مثال زیر را ببینید.

مثال ۲-۳. دو تراکنش T_1 و T_2 را به صورت زیر در نظر بگیرید و عملیاتی از این دو تراکنش بنویسید که منجر به بن بست شود.

T_1: $r_1[x] \rightarrow w_1[y] \rightarrow C_1$
T_3: $w_3[y] \rightarrow w_3[x] \rightarrow C_3$

عملیات زیر را در نظر بگیرید:

۱. در شروع هیچ‌یک از تراکنش‌ها قفلی را اخذ نکردند.

۲. زمان‌بند $r_1[x]$ را از مدیر تراکنش دریافت می‌کند، قفل $rl_1[x]$ را اخذ کرده (جدول قفل زیر) و $r_1[x]$ را به مدیر داده ارائه می‌کند.

قلم داده	نوع قفل	شماره تراکنش
X	r_l	T_1

۳. زمان‌بند $w_3[y]$ را از مدیر تراکنش دریافت می‌کند، قفل $wl_3[x]$ را اخذ کرده(جدول قفل زیر)، $w_3[y]$ را به مدیر داده ارسال می‌کند.

قلم داده	نوع قفل	شماره تراکنش
X	r_l	T_1
Y	w_l	T_3

۴. زمان‌بند $w_3[x]$ را از مدیر تراکنش دریافت می‌کند. زمان‌بند نمی‌تواند قفل $wl_3[x]$ را اخذ کند، زیرا با $rl_1[x]$ برخورد دارد. بنابراین، $w_3[x]$ به تأخیر می‌افتد. پس، آن را در صف قرار می‌دهد:

قلم داده	نوع قفل	شماره تراکنش منتظر
X	r_l	T_3

۵. زمان‌بند $w_1[y]$ را از مدیر تراکنش دریافت می‌کند، به همان دلیل بیان‌شده در مرحله 4، $w_1[y]$ باید به تأخیر بی‌افتد. پس، آن را در صف قرار می‌دهد:

قلم داده	نوع قفل	شماره تراکنش منتظر
X	r_l	T_3
Y	w_l	T_1

پس با توجه به قانون (۲) هیچ‌کدام از تراکنش‌های T_1 و T_3 نمی‌توانند کامل شوند. اگر زمان‌بند $w_1[y]$ را بدون اخذ قفل $wl_1[y]$ به مدیر داده بفرستد، قانون (۱) نقض می‌شود. به‌طور مشابه، برای $w_3[x]$ هم این مسئله رخ می‌دهد. فرض کنید زمان‌بند قفل $wl_3[y]$ را آزاد کند. بنابراین، می‌تواند قفل $wl_1[y]$ را اخذ کند و بنابراین اجازه دارد تا $w_1[y]$ را به مدیر داده بفرستد. در این حالت، زمان‌بند هرگز قادر نخواهد بود که قفل $wl_3[x]$ را اخذ کند (بنابراین، نمی‌تواند $w_3[x]$ را پردازش کند)، وگرنه، قانون (۳) نقض خواهد شد. به‌طور مشابه، اگر قفل $rl_1[x]$ را آزاد کند، بازهم قانون (۳) نقض خواهد شد.

این یک حالت **بن‌بست کلاسیک** است. درنتیجه، حتماً یکی از تراکنش‌های باید منبع(قفل) را آزادکرده و سقط گردد تا تراکنش دیگر بتواند پردازش شود.

گاهی بن‌بست زمانی رخ می‌دهد که تراکنش‌ها سعی دارند قفل‌های خواندن را به قفل‌های نوشتن تبدیل کنند. فرض کنید تراکنش T_i قلم داده x را می‌خواند و متعاقب آن سعی دارد تا در آن بنویسد. تراکنش T_i، $r_i[x]$ را به زمان‌بند می‌فرستد تا قفل $rl_i[x]$ را اخذ کند. وقتی تراکنش T_i، $w_i[x]$ را به زمان‌بند می‌فرستد، زمان‌بند باید $rl_i[x]$ را به $wl_i[x]$ ارتقاء دهد. این ارتقاء یک **تبدیل قفل** نام دارد. برای پیروی از 2PL زمان‌بند نباید $rl_i[x]$ را آزاد کند. چون قفل‌های اخذشده توسط یک تراکنش باهم برخوردی ندارند، در چنین مواقعی،

اگر دو تراکنش هم‌روند سعی در تبدیل قفل‌های خواندن‌شان به قفل نوشتن بر روی یک‌قلم داده را داشته باشند، منجر به بن‌بست می‌شود. مثال زیر را مشاهده کنید.

۱۳-۳. مسائل حل‌شده

مثال ۱. طرح اجرای زیر را در نظر بگیرید:

الف: تعیین کنید که آیا این طرح اجرا 2PL، S2PL و یا R2PL است یا خیر؟

ب: آیا طرح اجرا ترمیم‌پذیر است؟

ج: آیا طرح اجرا، اجتناب از سقوط آبشاری دارد یا خیر؟

تراکنش/ زمان	تراکنش T_1	تراکنش T_2
1	X (D)	
2	R (D)	
3	W (D)	
4		S (P)
5		R (P)
6	S (P)	
7	R (P)	
8		U (P)
9	R (D)	
10	U (D)	
11	C	
12		C

این طرح اجرا 2PL است، چون، در هر تراکنش اخذ قفل‌ها در یک مرحله و آزادسازی آن‌ها در مرحله دیگر انجام می‌شود. یعنی، پس از آزادسازی قفل برای هر تراکنش، دیگر قفلی اخذ نمی‌شود.

این طرح اجرا، S2PL است، چون آزادسازی قفل‌های از نوع X (انحصاری) به Commit آن تراکنش متصل است.

این طرح اجرا، R2PL نیست، چون در R2PL آزادسازی همه قفل‌ها باید به Commit آن تراکنش متصل است.

این طرح اجرا ترمیم‌پذیر و فاقد سقط آبشاری است، چون هیچ **خواندن از**[۱۲] انجام‌نشده است. یعنی، هیچ تراکنش مقدار قلمی از داده را پس از نوشتن توسط تراکنش دیگر نمی‌خواند.

مثال ۲. سه تراکنش زیر را در نظر بگیرید:

T_1: W(A), R(B), W(B), Commit.
T_2: R(A), R(B), Commit.
T_3: R(Z), R(A), Commit.

یک زمان‌بندی هم‌روند مثال بزنید که S2PL باشد، ولی R2PL نباشد.

طرح اجرای زیر را در نظر بگیرید.

تراکنش/ زمان	تراکنش T_1	تراکنش T_2	تراکنش T_3
1		S (A)	
2		S (B)	
3		R (A)	
4		U (A)	
5	X (A)		
6	W (A)		
7		R (B)	
8		U (B)	
9			S (Z)

[12].FROM READ

10			R (Z)
11	X (B)		
12	R (B)		
13	W (B)		
14	U (A)		
15	U (B)		
16	C		
17			S (A)
18			R (A)
19			U (A)
20			U (Z)
21			C
22		C	

این زمان‌بندی 2PL و S2PL است، چون آزادسازی قفل‌های نوع X (انحصاری) به COMMIT آن متصل است. این زمان‌بندی R2PL نیست، چون در تراکنش T_2 بین آزادسازی قفل‌های نوع S (اشتراکی) و COMMIT آن فاصله وجود دارد.

مثال ۳. آیا اجرا طرح زیر 2PL است؟

زمان/تراکنش	1	2	3	3	5	5
T_1	R (B)			R (A)		
T_2		R (B)	W (B)		R (A)	W (B)

این اجرا طرح 2PL است. اخذ قفل و آزادسازی آن‌ها در دو گام آمده است که در جدول زیر آمده است:

تراکنش/ زمان	تراکنش T_1	تراکنش T_2
1	S (B)	
2	S (A)	
3	R (B)	
4	U (B)	
5		X (B)
6		R (B)
7		W (B)
8	R (B)	
9	U (A)	
10		S (A)
11		R (A)
12		W (B)
13		U (B)
14		U (B)

چون اخذ قفل از پروتکل 2PL تبعیت می‌کند.

مثال ٤. زمان‌بند زیر را برای تراکنش‌های T_1 و T_2 در نظر بگیرید:

تراکنش/ زمان	T_1	T_2

1	R (A)	
2		R (B)
3	B:= B - 5	
4	W (B)	
5		R (A)
6	R (A)	
7		Display (A + B)
8	A := A + 50	
9	W (A)	
10	Display (A + B)	

اکنون به سؤالات زیر پاسخ دهید:

الف: گراف برخورد این زمان‌بند را رسم کنید؟

ب: آیا زمان‌بند پی‌درپی پذیر در برخورد است یا نه؟

ج: دستورات اخذ قفل و آزادسازی قفل برای تراکنش‌های T_1 و T_2 به‌طوری‌که آن‌ها از پروتکل قفل‌گذاری دومرحله‌ای (2PL) را رعایت کنند.

د: آیا زمان‌بند امکان دارد 2PL باشد یا خیر؟

گراف برخورد این زمان‌بند به شکل زیر است:

چون در این گراف دور وجود ندارد، پی‌درپی پذیر در برخورد است.

دستورات اخذ قفل و آزادسازی آن به‌صورت جدول زیر است:

T_1	T_2
X (B)	S (B)
R (B)	R (B)
B:= B-50	S (A)
W (B)	R (A)
X (A)	U (B)
R (A)	U (B)
A:=A +5	DISPLAY (A+B)
W (A)	
U (B)	
U (A)	
DISPLAY (A+B)	

زمان‌بند از پروتکل 2PL تبعیت نمی‌کند. زمان‌بند جزئی زیر را در نظر بگیرید:

T_1	T_2
X (B)	
R (B)	
	S (B)
	R (B)

چون T_1 یک قفل انحصاری (X-Lock) روی B نگه‌داری می‌کند، تراکنش T_2 باید تا زمانی که T_1 قفل را آزاد کند، منتظر بماند).

مثال ۵. زمان‌بند زیر را برای تراکنش‌های T_1 و T_2 در نظر بگیرید:

تراکنش / زمان	T_1	T_2
1	R (A)	
2		W (B)
3	R (B)	

آیا این زمان‌بند با پروتکل 2PL امکان‌پذیر است؟ اگر بله، دستورات اخذ و آزادسازی قفل را بنویسید؟

بله. مراحل اخذ و آزادسازی قفل به‌صورت زیر است.

تراکنش / زمان	1	2	3	4	5	6	7	8	9
T1	S(A)	R (A)				S(B)	R (B)	U (A)	U (B)
T_2			X(B)	W (B)	U (B)				

مثال ٦. تراکنش‌های زیر را در نظر بگیرید:

T_1: R (A)
R (B)
if A= 0 Then B:= B +1
W (B)

T_2: R (B)
R (A)
if B = 0 Then A = A + 1
W (B)

اکنون به سؤالات زیر پاسخ دهید:

الف: دستورات اخذ و آزادسازی قفل را برای تراکنش‌های T_1 و T_2 اضافه کنید، به‌طوری‌که پروتکل 2PL را رعایت کنند.

ب: یک زمان‌بند هم‌روند از T_1 و T_2 نشان دهید که بن‌بست داشته باشد. همچنین روند تکامل گراف انتظار برای (wait-for) را نشان دهید.

الف: دستورات اخذ و آزادسازی قفل به‌صورت زیر است:

T_1: S (A);
R (A);
X (B);
R (B);
if A = 0 Then B:= B+1
W (B);
U (A);
U (B);

T_2: S (B);
R (B)
X (A);
R (A);
if B= 0 Then A:=A+1
W (A);
U (B);
U (A);

در این زمان‌بند مطابق شکل زیر در گام ۶ بن‌بست رخ می‌دهد:

گام	تراکنش T_1	تراکنش T_2	گراف انتظار برای
1	S (A);		
2		S (B);	
3		R (B);	
4	R (A);		
5	X (B);		$T_1 \rightarrow T_2$
6		X (B);	$T1 \rightleftarrows T2$

مثال ۷. برای هر یک از زمان‌بندهای زیر به سؤالات ذیل پاسخ دهید:

الف: پی‌درپی پذیر هستند؟

ب: آیا می‌توانند با زمان‌بند قفل‌گذاری دومرحله‌ای (2PL) تولید شوند؟

ج: آیا می‌توانند با زمان‌بندی قفل‌گذاری دومرحله‌ای محض (S2PL) تولید شوند؟

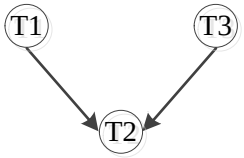
T1	W(A)		W(B)
T2		R(B)	R(D)
T3	R(D)	W(D)	

T1	W(A)		W(B)
T2		R(B)	R(D)
T3		W(D)	W(D)

T1	W(B)	W(B)	
T2		R(B)	
T3	R(D)		R(D)
			W(D)

	پی‌درپی پذیر	2PL	Strict 2PL
1	✓	✓	
2			
3	✓	✓	

زمان‌بند ۱، پی‌درپی پذیر است. چون گراف اولویت آن بدون دور است:

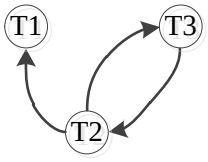


زمان‌بند ۱، 2PL است، چون قفل‌ها می‌توانند با توجه به جدول زیر تخصیص یابند:

T_1	T_2	T_3
		L(D) R(D)
L (A) R(A)		
	L(B) R(B)	
		W(D) U(D)
	L(B); U(B)	
L(B); W(B); U(B)		
	R(D); U(D)	

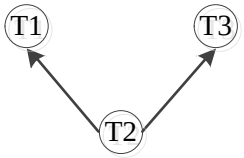
زمان‌بند ۱، 2PL محض نیست.

زمان‌بند ۲، پی‌درپی پذیر نیست، زیرا در گراف اولویت آن دور وجود دارد (گراف اولویت زیر):



هر زمان‌بند پی‌درپی پذیر نباشد، نمی‌تواند 2PL و 2PL محض باشد.

زمان‌بند ۳، پی‌درپی پذیر است. چون گراف اولویت آن دور وجود ندارد (گراف اولویت زیر):



با توجه به جدول ذیل 2PL است:

T_1	T_2	T_3
		S (D) R(D)
S (A) W (A)		
	S (B); S (D) R (D) U (B)	
S (B) W (B) U (A); U (B)		
	R (D) U (D)	
		X (D) W (D) U (D)

۱۴-۳. سؤالات چهارگزینه‌ای

۱. اگر تراکنش A بر روی تاپل t یک قفل اشتراکی (S) داشته باشد، پس:

الف: اگر از طرف تراکنش B یک قفل X بر روی تاپل t درخواست شود، این درخواست فوراً برآورده می‌شود.

ب: اگر از طرف تراکنش B یک قفل X بر روی تاپل t درخواست شود، این درخواست فوراً برآورده می‌گردد و تراکنش A سقوط می‌کند.

ج: اگر از طرف تراکنش B یک قفل S بر روی تاپل t درخواست شود، این درخواست فوراً برآورده می‌گردد و تراکنش A سقوط می‌کند.

د: اگر از طرف تراکنش B یک قفل S بر روی تاپل t درخواست شود، این درخواست فوراً برآورده می‌گردد.

۲. اگر تراکنش A قفل X (انحصاری) را بر روی چندتایی t قرار دهد آنگاه:

الف: تراکنش دیگری می‌تواند قفل X را بر روی چندتایی t بگذارد.

ب: تراکنش دیگری می‌تواند قفل S را بر روی چندتایی t بگذارد.

ج: هیچ تراکنشی نمی‌تواند بر روی چندتایی t قفل بگذارد.

د: تراکنش دیگری می‌تواند قفل X و S را بر روی چندتایی t بگذارد.

۳. در رابطه با قفل روی رابطه R در هنگام اجرای تراکنش T کدام‌یک از تعاریف زیر درست است؟

۱. IS: تراکنش T قفل S را روی تک‌تک تاپل‌های موجود در R برای پایداری آن تاپل‌ها در پردازش می‌زند.

۲. IX: همانند IS، به‌علاوه این‌که T می‌تواند تاپل‌های R را به‌طور انفرادی به هنگام کرده و روی آن‌ها قفل X بزند.
۳. S: T می‌تواند چندین خواندن را هم‌زمان انجام دهد ولی به هنگام‌سازی هم‌زمان امکان‌پذیر نیست.
٤. X: T می‌تواند چندین خواندن را هم‌زمان انجام دهد و تک‌تک تاپل‌ها را در R به هنگام کند.

الف: ۲ و ۳ و ۴ ب: ۱ و ۲ و ۳ ج: ۱ و ۲ و ۴ د: ۱ و ۳ و ۴

٤. با استفاده از قفل‌گذاری می‌توان سه مشکل هم‌روندی را حل نمود. اما متأسفانه قفل‌گذاری مشکلات خاص خود را دارد. کدام‌یک از موارد اساسی‌ترین آن مشکل‌ها محسوب می‌شود؟

الف: اجرای ناپیوسته ب: بن‌بست ج: انتظار د: بازگشت

۵. کدام‌یک از موارد زیر توصیف مناسب‌تری برای قفل IS می‌باشد؟ (فرض می‌کنیم تراکنش T قفلی بر روی رابطه R درخواست کرده است.)

الف: تراکنش T قصد دارد که قفل‌های S را روی چندتایی‌های منحصربه‌فردی از رابطه R بگذارد، جهت تضمین پایداری بر روی چندتایی‌های رابطه درحالی‌که آن‌ها پردازش می‌شوند.

ب: تراکنش T می‌تواند در رابطه R هم‌روندی تراکنش‌های خواننده (CONCURRENT READER) را تحمل کند، اما هم‌روندی تراکنش‌های به هنگام ساز (CONCURRENT UPDATERS) را تحمل نمی‌کند(خود تراکنش T نمی‌تواند هیچ چندتایی از رابطه R را به هنگام کند).

ج: تراکنش T در رابطه R می‌تواند هم‌روندی تراکنش خواننده را تحمل کند، اما هم‌روندی تراکنش‌های به هنگام ساز را تحمل نمی‌کند، به‌علاوه، تراکنش T ممکن است چندتایی‌های رابطه R را به هنگام کند و بنابراین قفل‌های X را روی این‌گونه چندتایی‌ها می‌گذارد.

د: تراکنش T به‌هیچ‌وجه نمی‌تواند هیچ دسترسی هم‌روند به رابطه R را تحمل کند (خود تراکنش T ممکن است چندتایی‌هایی از رابطه R را به هنگام سازد).

۱۵-۳. پاسخ تشریحی سؤالات چهارگزینه‌ای

۱. **گزینه (د) صحیح است.** چون اگر تراکنش A بر روی تاپل (چندتی) t یک قفل اشتراکی یا خواندن (S) را داشته باشد، دو حالت زیر ممکن است:

الف: اگر از طرف تراکنش دیگر B یک قفل انحصاری یا نوشتن (X) بر روی تاپل t درخواست گردد، این درخواست فوراً برآورده نمی‌شود.

ب: اگر از طرف تراکنش دیگر B یک قفل اشتراکی یا خواندن (S) بر روی تاپل t درخواست گردد، این درخواست می‌تواند فوراً برآورده شود.

۲. **گزینه (ج) صحیح است.** چون اگر تراکنش A بر روی تاپل t قفل انحصاری x (Exclusive lock) داشته باشد، هیچ قفلی از نوع انحصاری و اشتراکی نمی‌تواند به تراکنش مجزای B بر روی تاپل t اعطا شود.

۳. **گزینه (ب) صحیح است.** عبارت (د) نادرست است، چون، در قفل‌گذاری انحصاری (X) تراکنش T به‌هیچ‌وجه نمی‌تواند هیچ دسترسی هم‌زمان به رابطه R را تحمل کند.

۴. **گزینه (ب) صحیح است.** با استفاده از قفل‌گذاری می‌توان سه مشکل نتیجه ازدست‌رفته، وابستگی تثبیت نشده و تحلیل ناسازگاری را حل کرد، اما اساسی‌ترین مشکل قفل‌گذاری بن‌بست (Dead lock) می‌باشد.

بن‌بست موقعیتی است که در آن دو یا چند تراکنش هم‌روند در حال انتظار باشند و هر یک از آن‌ها منتظر باشند تا دیگری قفل را آزاد کند.

۵. **گزینه (الف) صحیح است.** قفل IS (Intent exclusive) که تعریف آن در گزینه (الف) آمده است. گزینه (ب) تعریف قفل S (اشتراکی) می‌باشد، گزینه (ج) تعریف قفل SIX (قفل اشتراکی قصدی انحصاری) است و گزینه (د) تعریف قفل انحصاری (X) می‌باشد.

فصل ۴ زمان‌بندهای غیر قفلی

۶.

۷. ۴-۱. مقدمه

۸. در این فصل، دو تکنیک زمان‌بندی ترتیب مهر زمانی (TO) و آزمون گراف پی‌درپی پذیری (SGT) که از قفل‌ها استفاده نمی‌کنند را شرح می‌دهیم. در این روش‌ها نیز مانند 2PL، نسخه‌های محافظه‌کارانه، مهاجم متمرکز و توزیع‌شده یا ترکیبی از این دو تکنیک وجود دارند.

۹. برخلاف قفل‌گذاری دومرحله‌ای 2PL، تکنیک‌های شرح داده‌شده در این فصل امروزه در بسیاری از محصولات تجاری استفاده نمی‌شوند. هرچند، کارایی آن‌ها نسبت به قفل‌گذاری دومرحله‌ای 2PL کاملاً شناخته نیست. بنابراین، اکثر مطالب این فصل در سطح مفهومی با جزئیات عملی و مقایسه کارایی کم‌تر نسبت به فصل سوم نشان داده‌شده است.

۱۰. ۴-۲. ترتیب مهر زمانی

۱۱. مقدمه

۱۲. درروش ترتیب مهر زمانی[۱۳]، مدیر تراکنش (TM) به هر تراکنش T_i مهر زمانی یکتا (منحصربه‌فردی) به نام $TS(T_i)$ تخصیص می‌دهد. تولید مهر زمانی همانند تکنیک‌هایی است که در فصل سوم بیان‌شده است. مدیر تراکنش (TM) به هر عمل که توسط تراکنش T_i تولید می‌شود، مهر زمانی T_i را ضمیمه می‌کند. بنابراین، صحبت درباره مهر زمانی یک عمل $O_i[x]$ که به‌آسانی همان مهر زمانی تراکنشی است که باعث وقوع آن عملیات شده است، خواهد بود. یک زمان‌بند ترتیب مهر زمانی (TO)، عملیات برخورد دار را به ترتیب مهر زمانی تراکنش‌ها مرتب می‌کند. زمان‌بند ترتیب مهر زمانی از قانون TO به‌صورت زیر استفاده می‌کند:

۱۳. قانون مهر زمانی (TO): اگر $p_i[x]$ و $q_j[x]$ عملیات برخورد داری باشند، آنگاه مدیر تراکنش (TM)، عمل $p_i[x]$ را قبل از $q_j[x]$ اجرا می‌کند، اگر و فقط اگر $TS(T_j) < TS(T_i)$ باشد. $TS(T_i)$ مهر زمانی تراکنش T_i است.

۱٤. قضیه ۱-٤: اگر H سابقه‌ای باشد که اجرائی را ارائه دهد که توسط یک زمان‌بند ترتیب مهر زمانی (TO) تولیدشده باشد، آنگاه H پی‌درپی پذیر (SR) است.

---

[13]. Time Stamp

۱۵. **اثبات:** SG(H) را در نظر بگیرید. اگر $T_i \rightarrow T_j$ لبه‌ای در SG(H) باشد، آنگاه باید عملیات برخورد داری مانند $p_i[x]$ و $q_j[x]$ در H وجود داشته باشند، به‌طوری‌که $p_i[x] < q_j[x]$ باشد. یعنی، با قانون ترتیب مهر زمانی $TS(T_i) < TS(T_j)$ است. اگر یک حلقه مانند $T1 \rightarrow Tn \rightarrow \ldots \rightarrow T2 \rightarrow T1$ در SG(H) وجود داشته باشد، آنگاه از طریق استنتاج $TS(T_1) < TS(T_1)$ را داریم که این امکان‌پذیر نیست. بنابراین، SG(H) بدون دور (غیر حلقوی) است و توسط قضیه پی‌درپی پذیری، سابقه H، پی‌درپی پذیری (SR) است.

۱۶. با اعمال قانون ترتیب مهر زمانی (TO) مطمئن می‌شویم که هر زوج از عملیات برخورد دار به ترتیب مهر زمانی‌شان اجرا خواهند شد. پس، یک اجرای ترتیب مهر زمانی همان اثری را خواهد داشت که یک اجرای پی‌درپی دارد. در ادامه این بخش روش‌های اعمال قانون ترتیب مهر زمانی را نشان می‌دهیم.

۱۷. ۳-۴. ترتیب مهر زمانی پایه‌ای

۱۸. ترتیب مهر زمانی پایه‌ای، پیاده‌سازی ساده و مهاجمانه‌ای از قانون ترتیب مهر زمانی است. این روش، عملیات را از مدیر تراکنش (TM) دریافت کرده و بلافاصله آن‌ها را به ترتیب ورود (اولین ورودی- اولین سرویس داده‌شده) به مدیر داده (DM) می‌فرستد.

۱۹. زمان‌بند برای اطمینان از سازگاری قانون ترتیب مهر زمانی، عملیاتی که خیلی دیر برسند، را رد می‌کند. عمل $p_i[x]$ خیلی دیر رسیده است، اگر بعد از خروج عملیات برخورد داری مانند $q_j[x]$ با $TS(T_j) > TS(T_i)$ رسیده باشد. اگر $p_i[x]$ خیلی دیر برسد، زمان‌بند نمی‌تواند بدون نقض قانون ترتیب مهر زمانی قبلاً خروجی $q_j[x]$ را داشته باشد، پس تنها راه‌حل رد کردن $p_i[x]$ است.

۲۰. اگر $p_i[x]$ رد شود، آنگاه T_i باید سقط شود. وقتی T_i مجدداً ارائه می‌گردد باید مهر زمانی بزرگ‌تری داشته باشد (به‌اندازه کافی بزرگ تا احتمال رد شدن آن در اجرای دوم کم‌تر باشد). به تفاوت بین اجتناب از بن‌بست مبتنی بر مهر زمانی و زمان‌بندی ترتیب مهر زمانی دقت کنید. در آنجا وقتی تراکنش سقط می‌گردید، با همان مهر زمانی قبلی دوباره ارائه می‌شد تا از حلقه در اجرای مجدد اجتناب کند. اما در اینجا، یک مهر زمانی جدید و بزرگ‌تر به آن تخصیص می‌یابد تا از رد کردن اجتناب کند.

۲۱. برای تعیین این‌که آیا عملی خیلی دیر رسیده است، زمان‌بند ترتیب مهر زمانی پایه‌ای، برای هر قلم داده x **حداکثر مهر زمانی خواندن** (R-TS(x) یا max-r-scheduled[x]) و **حداکثر مهر زمانی نوشتن** (W-TS(x) یا max-w-scheduled[x]) روی x را که به مدیر داده (DM) فرستاده است، نگه‌داری می‌کند. وقتی $p_i[x]$ به زمان‌بند برسد، به ازای تمام عملیات q که با $p_i[x]$ برخورد دارند، $TS(T_i)$ با max-q-scheduled[x] (q-ts(x)) مقایسه می‌نماید. اگر $TS(T_i)$ کوچک‌تر از max-q-scheduled[x] (q-ts(x)) باشد، آنگاه $p_i[x]$ را رد می‌کند، چون‌که قبلاً عمل برخورد داری با مهر زمانی بزرگ‌تر زمان‌بندی‌شده بود وگرنه زمان‌بند $p_i[x]$ را زمان‌بندی می‌کند، اگر $TS(T_i)$ بزرگ‌تر از max-p-scheduled[x] (p-ts(x)) باشد، آنگاه زمان‌بند max-p-scheduled[x] (p-ts(x)) را با $TS(T_i)$ به‌روز می‌کند.

۲۲. زمان‌بند با هماهنگی مدیر داده باید تضمین کند که مدیر داده عملیات را به همان ترتیبی که زمان‌بند ارسال می‌کند، اجرا می‌نماید. حتی اگر زمان‌بند تصمیم بگیرد که $p_i[x]$ می‌تواند زمان‌بندی شود، تا زمانی که همه $q_i[x]$ برخورد داری که قبلاً به مدیر داده ارسال‌شده‌اند، تأیید نشده باشند، نباید آن‌ها به مدیر داده برسند.

توجه کنید که 2PL (قفل‌گذاری دومرحله‌ای) به‌طور خودکار این عمل را انجام می‌دهد. چون قفل‌گذاری دومرحله‌ای تا زمانی که تمام عملیات برخورد دار قبلی زمان‌بندی‌شده قفل‌هایشان را آزاد نکنند، عملی را زمان‌بندی نمی‌کند.

۲۳. ۹–۴. مسائل حل‌شده

۲٤. مثال ۱. اگر زمان‌بندهای زیر که با یک زمان‌بندی مبتنی بر مهر زمانی (Timestamp – based) نشان داده شوند، کدام‌یک از تراکنش‌ها سقط (Abort) می‌شوند:

1) $r_1(x), w_1(x), r_2(z), r_1(y), w_1(y), r_2(x), w_2(x), w_2(z)$
2) $r_1(x), w_1(x), w_3(x), r_2(y), r_3(y), w_3(y), w_1(y), r_2(x)$
3) $r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), w_3(y), w_3(x), w_1(y), w_5(x), w_1(z), w_5(y), r_5(z)$
4) $r_1(x), r_3(y), w_1(y), w4(x), w_1(t), w5(x), r_2(z), r_3(z), w_2(z), w_5(z), r_4(t), r_5(t)$
5) $r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), r_3(y), r_3(x), w_1(y), w_5(x), w_1(z), r_5(y), r_5(z)$
6) $r_1(x), r_1(t), r_3(z), r_4(z), w_2(z), r_4(x), r_3(x), w_4(x), w_4(y), w_3(y), w_1(y), w_2(t)$
7) $r_1(x), r_4(x), w_4(x), r1(y), r_4(z), w_4(z), w_3(y), w_3(z), w_1(t), w_2(z), w_2(t)$

۲۵. زمان‌بندی گزینه (۱)، یعنی:

$r_1(x), w_1(x), r_2(z), r_1(y), w_1(y), r_2(x), w_2(x), w_2(z)$

۲۶. به‌صورت جدول زیر می‌باشد:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (X) = 1
write (x, 1)	OK	W-TS (X) = 1
read (z, 1)	OK	R-TS (Z) = 2
read (y, 1)	OK	R-TS (Y) = 1
write (y, 1)	OK	W-TS (Y) = 1
read (x, 2)	OK	R-TS (X) = 2
write (z, 2)	OK	W-TS (Z) = 2

۲۷. پس هیچ تراکنشی سقط نمی‌شود.

۲۸. زمان‌بندی گزینه (۲)، یعنی:

$r_1(x), w_1(x), w_3(x), r_2(y), r_3(y), w_3(y), w_1(y), r_2(x)$

۲۹. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (X) = 1
write (x, 1)	OK	W-TS (X) = 1
write (x, 3)	OK	W-TS (X) = 3
read (y, 2)	OK	R-TS (Y) = 2
read (y, 3)	OK	R-TS (Y) = 3
write (y, 3)	OK	W-TS (Y) = 3
write (y, 1)	تراکنش T_1 سقط شد	———
read (x, 2)	تراکنش T_2 سقط گردید	———

۳۰. زمان‌بندی گزینه (۳)، یعنی:

$r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), w_3(y), w_3(x), w_1(y), w_5(x), w_1(z), w_5(y), r_5(z)$

۳۱. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (x) = 1

read (x, 2)	OK	R-TS (x) = 2
write (x, 2)	OK	W-TS (x) = 2
read (x, 3)	OK	R-TS (x) = 3
read (z, 4)	OK	R-TS (z) = 4
write (x, 1)	تراکنش T_1 سقط می‌شود	———
write (y, 3)	OK	W-TS (y) = 3
write (x, 5)	OK	W-TS (x) = 5
write (y, 5)	OK	W-TS (y) = 5
read (z, 5)	OK	R-TS (z) = 5

۳۲. زمان‌بندی گزینه (۴)، یعنی:

$r_1(x), r_3(y), w_1(y), w4(x), w_1(t), w5(x), r_2(z), r_3(z), w_2(z), w_5(z), r_4(t), r_5(t)$

۳۳. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (x) = 1
read (y, 3)	OK	R-TS (y) = 3
write (y, 1)	تراکنش T_1 سقط گردید	———
write (x, 4)	OK	W-TS (x) = 4
write (x, 5)	OK	W-TS (x) = 5
read (z, 2)	OK	R-TS (z) = 2
read (z, 3)	OK	R-TS (z) = 3
write (z, 2)	تراکنش T_2 سقط شد	———
write (z, 5)	OK	W-TS (z) = 5
read (t, 4)	OK	R-TS (t) = 4
read (t, 5)	OK	R-TS (t) = 5

۳۴. زمان‌بندی گزینه (۵)، یعنی:

$r_1(x), r_2(x), w_2(x), r_3(x), r_4(z), w_1(x), r_3(y), r_3(x), w_1(y), w_5(x), w_1(z), r_5(y), r_5(z)$

۳۵. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (x) = 1
read (x, 2)	OK	R-TS (x) = 2
read (x, 3)	OK	R-TS (x) = 3
read (z, 4)	OK	R-TS (z) = 4
write (x, 1)	تراکنش T_1 سقط شد	———
read (y, 3)	OK	R-TS (y) = 3
read (x, 3)	OK	R-TS (x) = 3
write (x, 5)	OK	W-TS (x) = 5
read (y, 5)	OK	R-TS (y) = 5
read (z, 5)	OK	R-TS (z) = 5

۳۶. جدول زمان‌بندی گزینه (۶)، یعنی:

$r_1(x), r_1(t), r_3(z), r_4(z), w_2(z), r_4(x), r_3(x), w_4(x), w_4(y), w_3(y), w_1(y), w_2(t)$

۳۷. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (x) = 1
read (t, 1)	OK	R-TS (t) = 1
read (z, 3)	OK	R-TS (z) = 3

read (z, 4)	OK	R-TS (z) = 4
write (z, 2)	تراکنش T_2 سقط شد	———
read (x, 4)	OK	R-TS (x) = 4
read (x, 3)	تراکنش T_3 سقط شد	———
write (x, 4)	OK	W-TS (x) = 4
write (y, 2)	OK	W-TS (y) = 2
write (y, 2)	تراکنش T_1 سقط شد	———
write (t, 2)	OK	W-TS (t) = 2

۳۸. جدول زمان‌بندی گزینه (۷)، یعنی:

$r_1(x), r_4(x), w_4(x), r1(y), r_4(z), w_4(z), w_3(y), w_3(z), w_1(t), w_2(z), w_2(t)$

۳۹. در جدول زیر آمده است:

عمل	پاسخ	مقادیر جدید
read (x, 1)	OK	R-TS (x) = 1
read (x, 4)	OK	R-TS (x) = 4
write (x, 4)	OK	W-TS (x) = 4
read (y, 1)	OK	R-TS (y) = 1
read (z, 4)	OK	R-TS (z) = 4
write (y, 3)	OK	W-TS (y) = 3
write (z, 3)	تراکنش T_3 سقط گردید	———
write (t, 1)	OK	W-TS (t) = 1
write (z, 2)	تراکنش T_2 سقط گردید	———

۱۰-۴. سؤالات چهارگزینه‌ای

۱. در پروتکل TO بین دو عمل متعارض همیشه عمل ..... رد می‌شود.

الف: متقدم ب: متأخر ج: متأخر، اگر نوشتن باشد. د: متقدم، اگر خواندن باشد.

۲. در الگوریتم WD اگر تراکنش T_i خواهان قفل باشد و تراکنش T_j روی‌داده موردنیاز T_i قفل داشته باشد، در چه صورت T_i طرد می‌شود؟

الف: $TS(Ti) > TS(Tj)$ ب: $TS(Ti) < TS(Tj)$

ج: $TS(Ti) \geq TS(Tj)$ د: $TS(Ti) \leq TS(Tj)$

۳. کدام‌یک از روش‌های زیر برای حل مسئله بن‌بست می‌تواند منجر به مسئله قطعی (Starvation) شود؟

الف: مکانیسم wait-Die ب:مکانیسم wound-wait

ج:مکانیسم پایان زمانی Timeout د: هیچ‌کدام

٤. کدام عبارت در مورد جلوگیری از بن‌بست صحیح است؟

الف: در روش wound – wait تراکنشی که مجدداً شروع می‌شود، مهر زمانی جدید می‌گیرد.

ب: در روش wait - Die تراکنشی که مجدداً شروع می‌شود، مهر زمانی جدید می‌گیرد.

ج: در روش wait-Die تراکنشی که مجدداً شروع می‌شود، مهر زمانی اصلی خود را حفظ می‌کند.

د: در روش wound – wait تراکنشی که مجدداً شروع می‌شود، ممکن است دچار Livelock بشود.

۱۱-۴. پاسخ تشریحی سؤالات چهارگزینه‌ای

۱. **گزینه (ب) صحیح است.** در پروتکل مهر زمانی مبنایی، اگر ترتیب اجرای تراکنش‌ها بر اساس مهر زمانی شان رعایت نشود، تراکنش T سقط می‌گردد و دوباره با مقدار مهره زمانی جدید از سر گرفته می‌شود. برای تضمین رعایت مهر زمان در اجرای دو دستور متعارض از پروتکلی به نام TO استفاده می‌شود. هدف اصلی این پروتکل مرتب کردن تراکنش‌ها به صورتی است که در صورت به‌روز تعارض در عملیات، تراکنش قدیمی‌تر اولویت اجرا دارد. یعنی، در پروتکل TO بین دو عمل متعارض همیشه عمل متأخر سقط می‌گردد.

۲. **گزینه (ج) صحیح است.** اگر تراکنشی دچار بن‌بست شود، مرتباً به‌عنوان قربانی انتخاب‌شده و سقط می‌گردد و هرگز اجرای آن خاتمه نمی‌یابد. برای رفع این مشکل الگوریتم‌های WD(Wait-Die) و WW(Wound Wait) معرفی‌شده‌اند. در الگوریتم WD، اگر $TS(T_j) < TS(T_i)$ باشد، آنگاه T_i قفل روی‌داده را می‌خواهد، منتظر می‌ماند تا T_j خاتمه یابد. وگرنه، T_i سقط می‌شود تا دیرتر با همان مقدار مهر زمانی مجدداً شروع کند. اما در الگوریتم WW، اگر $TS(T_i) < TS(T_j)$ باشد و T_j روی‌داده قفل داشته باشد، آنگاه T_j ساقط می‌شود. در این الگوریتم حق تقدم زمانی رعایت می‌گردد. اما در الگوریتم WD حق تقدم زمانی رعایت نمی‌شود.

۳. **گزینه (ج) صحیح است.** درروش Wait-Die، تراکنش پیرتر (با برچسب زمانی کم‌تر)، منتظر تراکنش جوان‌تر می‌ماند تا قفل‌هایش را آزاد کند. تراکنش جوان‌تر هرگز منتظر تراکنش پیرتر نمی‌ماند، بلکه سقط می‌شود (می‌میرد) و باعث می‌شود بن‌بست پیش نیاید. درروش WOUND – WAIT، تراکنش پیرتر به‌جای انتظار، تراکنش جوان‌تر را می‌کشد. تراکنش جوان‌تر منتظر تراکنش پیرتر می‌ماند. این روش ممکن است کم‌تر منجر به سقط شدن تراکنش‌ها شود. در این روش گرسنگی (STRAVATION) به وجود نمی‌آید. اما در طرح مکانیسم فرصت زمانی، یک تراکنش برای مدت معینی منتظر می‌شود تا یک قفل را اخذ کند. پس از سپری شدن آن زمان انتظار تراکنش عقب‌گرد (ROLLBACK) می‌شود. بنابراین، در این روش امکان به وجود آمدن بن‌بست وجود ندارد. پیاده‌سازی این روش ساده است، البته امکان ایجاد گرسنگی وجود دارد. چون، امکان عقب‌گرد و سقط شدن یک تراکنش با اولویت پایین وجود دارد. یکی از مشکلات این روش دشوار بودن مشخص نمودن زمان انتظار مناسب تراکنش‌ها است.

۴. **گزینه (ج) صحیح است.** باید دقت کنید که در روش‌های wound-wait و Wait-Die، تراکنش مجدداً اجرا شود، با مهر زمانی اولیه (اصلی‌اش) اجرا می‌گردد. در این روش‌ها بن‌بست و گرسنگی (قحطی)رخ نمی‌دهد.

فصل ۵ ترمیم متمرکز

۱–۵. خرابی‌ها

سؤالی که در این فصل مطرح می‌گردد این است که چگونه می‌توان تراکنش‌ها را پردازش کرد تا بتوانند خطاها را تحمل نمایند. در این فصل، ترمیم متمرکز سیستم‌های پایگاه داده را بحث می‌کنیم تا بتوانیم به این

سوال پاسخ دهیم. برای این منظور، ابتدا انواع خرابی‌ها را بیان می‌نماییم. این واقع‌بینانه نیست که انتظار ایجاد سیستم‌های پایگاه داده داشته باشیم که بتوانند انواع خرابی‌های احتمالی را تحمل کنند. بااین‌حال، سیستم خوب باید بتواند به‌طور خودکار رایج‌ترین انواع خرابی‌ها را بدون دخالت انسان تحمل کند. بسیاری از خرابی‌ها ناشی از ورود داده نادرست و اشتباه در برنامه‌نویسی تراکنش‌ها است (تعیین نادرست پارامترهای تراکنش‌ها). متأسفانه این خرابی‌ها موجب می‌شوند تا سازگاری پایگاه داده پس از اجرای تراکنش‌ها حفظ نشود.

بسیاری از خرابی‌ها ناشی از اشتباه کاربران (اپراتورها) است. به‌عنوان‌مثال، یک اپراتور ممکن است دستوری را به‌اشتباه تایپ کرده، و بخشی از داده‌های پایگاه داده آسیب ببیند، یا موجب می‌شود که رایانه در هنگام اجرای یک تراکنش راه‌اندازی مجدد شود. به‌طور مشابه، یک تکنسین رایانه ممکن است به یک دیسک یا نوار در طی رویه نگه‌داری صدمه وارد کند. خطر ابتلا به این اشتباهات را می‌توان با بهره‌برداری طراحی مناسب رابط کاربر و آموزش اپراتورها بهبود ببخشید. جلوگیری از این اشتباهات خارج از حوزه ترمیم سیستم‌های پایگاه داده است. هرچند مکانیسم‌هایی برای ترمیم سیستم پایگاه داده طراحی‌شده‌اند تا با برخی از پیامدهای این اشتباهات مقابله کنند.

با توجه به این فرضیات، سه نوع خرابی در سیستم مدیریت پایگاه داده (DBMS) متمرکز بسیار مهم است که عبارت‌اند از:

۱. **خرابی تراکنش**، وقتی رخ می‌دهد که تراکنش سقط می‌شود. این نوع خرابی به دودسته خطای منطقی و خطای سیستمی تقسیم می‌شود. **خطای منطقی**، به دلیل برخی اشکالات نظیر ورودی‌های بد، پیدا نکردن داده و تمام نشدن تراکنش به وجود می‌آید و **خطای سیستمی**، حالتی است که در آن سیستم به دلیل بن‌بست و غیره به وضعیت نامطلوب می‌رود. بنابراین، دیگر تراکنش‌ها در حالت عادی قابل‌اجرا نیستند.

۲. **خرابی سیستمی**، به خرابی یا از دست رفتن محتویات حافظه فرار (مانند حافظه اصلی) اشاره دارد. به‌عنوان‌مثال، این خرابی‌ها می‌توانند به دلیل قطع برق، خرابی و عملکرد بد سخت‌افزار و یا خطای سیستم‌عامل اتفاق بیافتد. خرابی سیستمی موجب می‌شود معمولاً یک یا تمام تراکنش‌های فعال آسیب ببینند، ولی، داده‌های ذخیره‌شده در حافظه‌های جانبی دچار خرابی نمی‌شوند.

۳. **خرابی رسانه**، وقتی رخ می‌دهد که بخشی از حافظه جانبی(دیسک) خراب می‌شود. به‌عنوان نمونه، این خرابی وقتی‌که تعدادی از سکتورهای دیسک خراب شود، رخ می‌دهد. این نوع خرابی روی تراکنش‌هایی که در حال استفاده از این بخش داده(بخش خراب‌شده) هستند، تأثیر می‌گذارد.

روش‌هائی که برای مقابله با خرابی رسانه مورداستفاده قرار می‌گیرند ازنظر مفهومی مشابه آن است که برای خرابی سیستمی به کار می‌رود. در هر دو حالت، فرض می‌کنیم که بخشی از حافظه غیرقابل‌اعتماد است. یعنی، حافظه فرار در حالت خرابی‌های سیستمی و بخشی از حافظه جانبی، در حالت خرابی‌های رسانه غیرقابل‌اعتماد می‌شوند. برای حفاظت داده‌ها در حافظه غیرقابل‌اعتماد، معمولاً کپی دیگری از داده در محل متفاوتی از این حافظه نگه‌داری می‌شود.

۲-۵. انواع حافظه‌های ذخیره‌سازی

از جهت قابلیت نگه‌داری اطلاعات در صورت وقوع خرابی، حافظه‌های ذخیره‌سازی به سه دسته تقسیم می‌شوند که عبارت‌اند از:

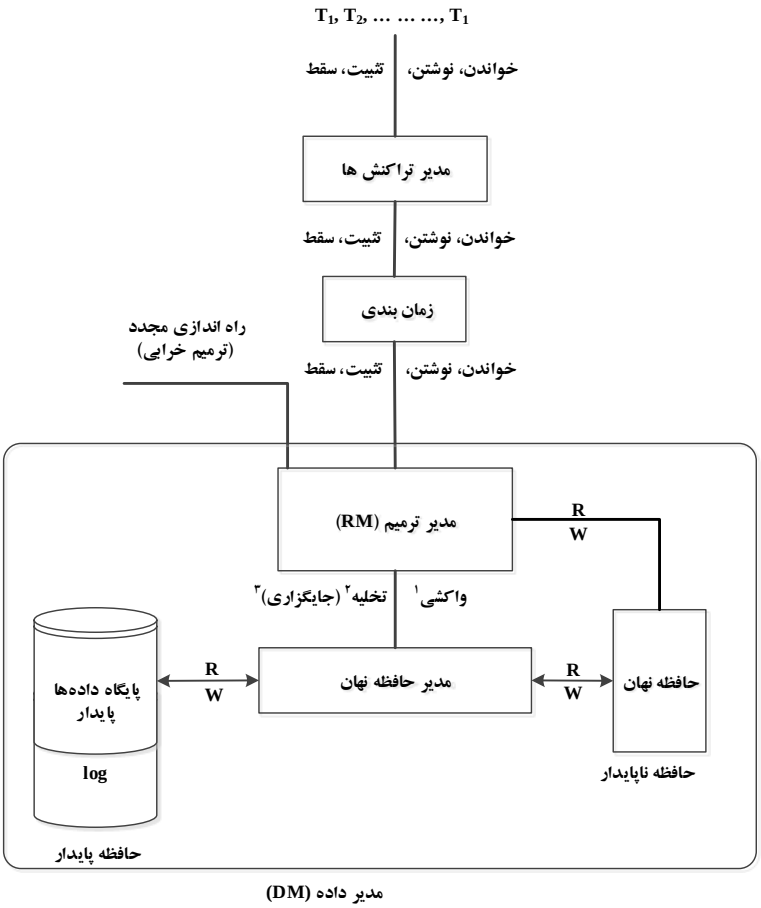
۱. **حافظه فرار،** حافظه‌ای که در صورت وقوع خرابی سیستم، اطلاعات آن از بین می‌رود، مثل حافظه اصلی و حافظه نهان.

۲. **حافظه جانبی یا کمکی (غیر فرار)،** خرابی سیستم را تحمل کرده و اطلاعات آن‌ها حتی با وقوع خرابی سیستم، قابل بازیابی می‌باشد. مانند دیسک‌ها، فلش‌ها، نوارهای مغناطیسی و غیره.

۳. **حافظه مانا (پایدار)**، رسانه‌ای که ایده آل ما است و در برابر تمام خرابی‌ها مصون می‌ماند. این نوع رسانه درواقع یک مفهوم منطقی است نه فیزیکی و هنوز چنین رسانه‌ای وجود خارجی ندارد، بلکه سعی می‌کنیم با راهکارهایی که از رسانه‌های فیزیکی موجود استفاده می‌کنند، به این هدف ایده آل نزدیک شویم. متداول‌ترین راهکار، استفاده از چندین کپی از داده‌ها روی رسانه‌های مختلف است. با این کار سعی می‌کنیم احتمال از دست رفتن اطلاعات در صورت وقوع خرابی را به صفر نزدیک‌تر کنیم.

۳-۵. معماری مدیر داده

در فصل اول مدلی شرح داده‌شده که از مولفه‌هائی مدیر تراکنش، زمان‌بند و مدیر داده تشکیل‌شده است. در فصل اول به‌طور مختصر درباره این مدل و مولفه‌هائی آن بحث گردید. در فصل‌های دوم، سوم و چهارم بر روی مدیر تراکنش و زمان‌بند تمرکز کردیم، در این فصل بر روی مدیر داده تمرکز می‌کنیم. با مدیر داده حافظه را مدیریت می‌کنیم. حافظه‌ای که در صورت بروز خرابی ممکن است بخش‌هائی از اطلاعات آن از بین برود. فرض می‌کنیم که خرابی‌های سیستمی منجر به از دست رفتن محتوی حافظه فرار می‌شود و محتوی حافظه پایدار از بین نمی‌رود. بنابراین، باید تفاوت بین حافظه فرار و حافظه پایدار را که به‌طور خلاصه در فصل اول بحث گردیده است، بدانیم. اجازه دهید به بررسی مجدد مدل سیستم پایگاه داده با تمرکز بر مسائل این فصل بپردازیم(شکل ۵-۱ را ببینید).



شکل ۱-۵ مدل یک سیستم پایگاه داده متمرکز

در این مدل، عملیات تراکنش‌ها به مدیر تراکنش ارسال می‌شوند و مدیر تراکنش آن‌ها را به زمان‌بند می‌فرستد. زمان‌بند چهار عمل Read، Write، Commit و Abort را بلافاصله به مدیر داده می‌فرستد. زمان‌بند، برای اجرای عملیات Read، Write و Commit باید تصمیم بگیرد که احتمالاً بعد از مقداری تأخیر عمل را بپذیرد یا رد کند. اگر انجام عمل را رد کند، باید یک تأییدیه منفی به مدیر تراکنش بفرستد. این تأییدیه منفی موجب می‌شود مدیر تراکنش یک عمل Abort به زمان‌بند بفرستد و زمان‌بند بلادرنگ آن را به مدیر داده می‌فرستد. اگر زمان‌بند عمل را بپذیرد، آن را به مدیر داده می‌فرستد که جزئیات تغییر در محتوی حافظه به الگوریتم مدیر داده بستگی دارد. جزئیات این تغییرات در محتوای حافظه و الگوریتم‌های مدیر داده موضوع اصلی این فصل می‌باشد. وقتی مدیر داده پردازش عمل را خاتمه دهد، تأییدیه خود را به زمان‌بند می‌فرستد تا زمان‌بند آن تأییدیه را به مدیر تراکنش ارسال کند. برای عمل خواندن، مقدار خوانده‌شده همان تأییدیه انجام عمل می‌باشد.

علاوه بر عملیات Read، Write، Commit و Abort، مدیر داده ممکن است عمل Restart را نیز دریافت کند. عمل Restart توسط ماژول خارجی نظیر سیستم‌عامل صادر می‌شود.

حاصل Restart رفتن پایگاه داده به حالت سازگار است، یعنی، اثرات تراکنش‌های تثبیت نشده را حذف می‌کند و اثرات گم‌شده تراکنش‌های تثبیت‌شده را اعمال می‌کند. به عبارت دقیق‌تر، منظور از آخرین مقدار تثبیت‌شده از قلم داده x در اجرا، مقداری است که برای آخرین بار توسط یک تراکنش تثبیت‌شده در x نوشته‌شده است. حالت تثبیت‌شده پایگاه داده وضعیتی است که در آن هر قلم داده آخرین مقدار تثبیت‌شده خودش را دارد. عمل Restart پایگاه داده را وضعیت تثبیت‌شده‌اش قبل از وقوع خرابی سیستم باز می‌گردد.

برای نشان دادن صحت، از تئوری‌های بیان‌شده فصل دوم استفاده می‌کنیم. فرض کنید H سابقه‌ای باشد که اعمال یک ترتیب جزئی از عملیات پردازش‌شده توسط مدیر داده (DM) تا زمان خرابی سیستم باشد. تصویر تثبیت‌شده H، یعنی C(H)، با حذف همه تراکنش‌های تثبیت نشده از H است. اگر H توسط یک زمان‌بند صحیح تولید گردیده باشد، آنگاه ترمیم‌پذیر است. مقادیر خوانده یا نوشته‌شده در C(H)، همان مقادیر خوانده یا نوشته‌شده توسط عملیات متناظر در H است. بنابراین، Restart با بازیابی آخرین مقدار تثبیت‌شده هر قلم داده، موجب می‌گردد تا پایگاه داده اجرای ارائه‌شده سابقه C(H) را ارائه نماید، یعنی، اجرای تراکنش‌هایی که در هنگام خرابی سیستم تثبیت‌شده بودند. به‌علاوه، C(H) پی‌درپی پذیر (SR) است، زیرا، توسط یک زمان‌بند صحیح تولیدشده بود.

بنابراین، وقتی‌که Restart خاتمه می‌یابد، پایگاه داده در یک وضعیت سازگار است. هدف اصلی این فصل، شرح ساختمان‌های داده‌ای است که باید توسط مدیر داده (DM) نگه‌داری شود تا این‌که Restart بتواند فقط با استفاده از اطلاعات ذخیره‌شده در حافظه پایدار، ترمیم را انجام دهد.

۴-۵. حافظه پایدار

مدیر داده (DM) از دو بخش تشکیل‌شده است:

۱. **مدیر حافظه نهان (CM)**، که حافظه را دست‌کاری می‌کند.

۲. **مدیر ترمیم**، که عملیات Read، Write، Commit، Abort و Restart را پردازش می‌کند.

مدیریت حافظه نهان (CM) هم واکشی (Fetch) داده از حافظه پایدار به حافظه فرار را انجام می‌دهد و هم عمل فلاش (Flush) جهت انتقال داده از حافظه فرار به حافظه پایدار را انجام می‌دهد. مدیر ترمیم کنترل جزئی روی عملیات فلاش مدیر حافظه نهان انجام می‌دهد تا مطمئن گردد که حافظه پایدار همیشه شامل داده‌هائی است که مدیر ترمیم با استفاده از آن‌ها، عمل Restart را به‌درستی انجام دهد.

فرض می‌کنیم که زمانی مدیر حافظه نهان دستور نوشتن یک‌قلم داده در حافظه پایدار را صادر می‌کند، آن عمل یا یکجا انجام می‌شود یا اصلاً انجام نمی‌شود و نتیجه با یک مقدار بولین به اطلاع ماژول فراخواننده می‌رسد یا حتی اگر قبل از نوشتن روی دیسک، خرابی سیستمی رخ دهد، چنین نوشتن‌هایی **اتمیک** نامیده می‌شوند. اگر اتمیک بودن عمل نقض شود (از قبیل تغییر برخی داده‌ها نه همه آن‌ها)، حتماً یک خرابی رسانه اتفاق افتاده است. هرچند فعلاً فرض می‌کنیم که خرابی رسانه اتفاق نمی‌افتد. یعنی، همه نوشتن‌ها اتمیک هستند.

در حافظه کمکی (یعنی دیسک‌ها)، واحد نوشتن معمولاً صفحه با اندازه ثابت (یا بلاک) است. وقتی‌که یک بلاک روی دیسک نوشته می‌شود، دو نتیجه ممکن است رخ دهد:

۱. بلاک به‌درستی بازنویسی شده است و تا بازنویسی بعدی، حالت جدید خود را نگه می‌دارد.
۲. مقدار جدید بلاک طوری از بین می‌رود که وقتی بلاک بعدی خوانده می‌شود، حالت خطا تشخیص داده می‌شود.

یعنی، یا عمل به‌درستی اجرا می‌شود یا با خرابی رسانه مواجه می‌گردد. تشخیص خطا به‌طور طبیعی توسط سخت‌افزار دیسک پشتیبانی می‌شود. اگر تعداد کمی از بیت‌های خطا مربوط به محتوی صفحه تغییر کرده باشد، سخت‌افزار دیسک خطا را از طریق محاسبه Checksum در هنگام خواندن صفحه تشخیص می‌دهد.

در عمل، واحد داده قابل‌نوشتن که به مدیر داده ارسال می‌گردد، با واحد نوشتن اتمیک روی حافظه کمکی می‌تواند متفاوت باشد. یعنی، واحد نوشتن مدیر داده ممکن است در حد رکوردهای کوتاه (خیلی کوتاه‌تر از یک صفحه) باشد، ولی واحد نوشتن اتمیک روی دیسک در قالب صفحه است. این عدم تطابق واحد قلم داده، زمان طراحی الگوریتم‌های ترمیم، توجه ویژه‌ای نیاز دارد.

در این فصل، برای سادگی، فرض می‌کنیم که واحد داده مدیر داده (DM)، هم‌اندازه واحد داده‌ای است که توسط حافظه پایدار پشتیبانی می‌شود. یعنی، با تکنولوژی دیسک‌های امروزی، قلم داده همان صفحه با اندازه ثابت است.

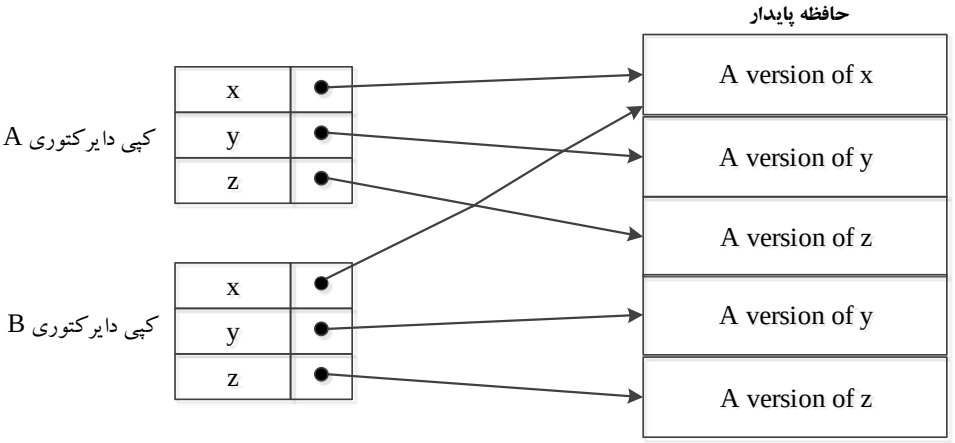
مدیران داده می‌توانند فقط یک کپی یا بیش یک کپی از هر قلم داده در حافظه پایدار نگه داری کنند. بین مدیران داده که فقط یک کپی از هر قلم داده را در حافظه پایدار نگه می‌دارند و مدیران داده‌هائی که بیش از یک کپی از هر قلم داده را نگه‌داری می‌کنند، تفاوت وجود دارد. هرگاه فقط یک کپی از قلم داده وجود داشته باشد، آنگاه با هر بازنویسی قلم داده، مقدار قبلی آن از بین می‌رود. این روش **تغییر درجا** نامیده می‌شود. اگر بیش از یک کپی از هر قلم داده وجود داشته باشد، مدیر حافظه نهان می‌تواند یک‌قلم داده را در حافظه پایدار بنویسد، بدون این‌که نسخه‌های قدیمی‌تر قلم داده از بین برود. نسخه‌های قدیمی‌تر کپی‌های **سایه** (Shadow) نامیده می‌شود و این روش **سایه‌زنی** (Shadowing) نامیده می‌شود.

با روش Shadowing، نگاشت اقلام داده‌ای به مکان‌های حافظه پایدار در مدت‌زمان تغییرات انجام می‌شود. بنابراین، لازم است فهرستی برای پیاده‌سازی این نگاشت وجود داشته باشد که به ازای هر قلم داده و ورودی، یک سطر داشته باشد که در آن قلم داده و مکان حافظه پایدارش ثبت گردد (شکل ۲ -۵ را ببینید).

معمولاً به ازای هر تعداد حالت متفاوت از پایگاه داده، همان تعداد نسخه‌ها، چنین فهرستی وجود دارد. حالت پایگاه داده در حافظه پایدار را **پایگاه داده پایدار** تعریف می‌کنیم. با روش درجا تنها یک نسخه از هر قلم داده در حافظه پایدار وجود دارد. با روش Shadowing، فهرست خاصی به نام D در حافظه پایدار وجود دارد که حالت فعلی پایگاه داده را تعریف می‌کند. کپی‌های اقلام داده‌ی در حافظه پایدار که توسط فهرست‌هایی غیر از D ارجاع داده می‌شوند را **کپی‌های سایه** (Shadow) می‌نامند.

۵-۵. مدیریت حافظه نهان

برای اجتناب از دسترسی به حافظه پایدار به ازای هر پردازش خواندن و نوشتن، بهتر است که کپی از پایگاه داده در حافظه فرار نگه‌داری شود. در این صورت، برای خواندن یک‌قلم داده کافی است کپی موجود



شکل ۲-۵ مثالی از سایه زنی با استفاده از سایه زنی.

در حافظه فرار دست‌یابی شود. برای نوشتن یک‌قلم داده، باید مقدار جدید هم در حافظه فرار و هم در حافظه پایدار نوشته شود. کپی موجود در حافظه پایدار فقط برای ترمیم از خرابی‌های سیستمی مفید است. ازآنجایی‌که دسترسی به حافظه پایدار کندتر از دسترسی به حافظه فرار است، در اکثر موارد کپی پایگاه داده در حافظه فرار کارایی را بهبود می‌بخشد. متأسفانه، نگه‌داری یک کپی کامل داده‌ها در حافظه فرار به دلیل اندازه بزرگ پایگاه داده و قیمت بالای این نوع حافظه ممکن نیست. هرچند، با ارزان شدن حافظه فرار ممکن است این روش در آینده متداول‌تر شود.

درهرصورت، در حال حاضر باید بخشی از پایگاه داده(کم‌تر از کل پایگاه داده) را در حافظه فرار نگه‌داریم. این عمل می‌تواند شبیه کش کردن سخت‌افزار و حافظه مجازی سیستم‌عامل توسط روشی که **نهان کردن یا بافرینگ** نامیده می‌شود، انجام شود. یعنی، حافظه نهان برای نگه‌داری بخش‌هائی از پایگاه داده اختصاص می‌یابد. **حافظه نهان**، مجموعه‌ای از اسلات‌هایی است که هر اسلات می‌تواند یک‌قلم داده را در خود ذخیره کند (شکل ۳-۵ را ببینید). هر اسلات به‌اندازه واحد قلم ذخیره‌شده است که می‌تواند به‌طور اتمیک در حافظه پایدار در قالب واحد صفحه نوشته شود.

ترافیک اقلام داده‌ای به داخل و خارج حافظه نهان توسط مدیر حافظه نهان (CM) با Flush و Fetch کنترل می‌شود. عمل Flush، اسلات C از حافظه نهان را به‌عنوان پارامتر می‌گیرد و اگر C اسلات تغییرنیافته باشد که از طریق فیلدdirty قابل تشخیص است، آنگاه Flush هیچ کاری انجام نمی‌دهد. اما، اگر C تغییریافته باشد، مقدارش را در محل قلم داده ذخیره‌شده در C می‌نویسد و فیلد dirty آن را ریست می‌کند. Flush، تا زمانی که به‌روزرسانی C روی حافظه پایدار کامل نشده است، به ماژول فراخوان خودش چیزی برنمی‌گرداند.

توجه داشته باشید که نگاشت بین هر قلم داده و مکان نگه‌داری آن باید روی حافظه پایدار نگه‌داری شود. اگر مدیر حافظه نهان از Shadowing استفاده کند، آنگاه این نگاشت به کمک یک فهرست انجام می‌شود. مدیر حافظه نهان بسته به الگوریتم ترمیم فهرست‌ها را انتخاب و دست‌کاری می‌کند. اگر مدیر حافظه نهان از تغییر درجا استفاده کند، آنگاه این نگاشت یکتا است.

دایرکتوری نهان

شماره اسلات	نام قلم داده
2	x
1	y . . .

نهان

شماره اسلات	بیت کثیف	مقدار قلم داده
1	1	“October 12”
2	0 . . .	3.1416

شکل ۳-۵ ساختار کش.

عمل Fetch، نام قلم داده x را به‌عنوان پارامتر دریافت کرده، مدیر حافظه نهان را مجبور می‌نماید تا اعمال زیر را انجام دهد:

۱. یک اسلات خالی به نام C را در حافظه نهان انتخاب می‌کند. اگر تمام اسلات‌های حافظه نهان پر باشند، اسلات C را انتخاب می‌کند و محتوی آن را بر روی حافظه پایدار می‌نویسد و سپس C را به‌عنوان اسلات خالی استفاده می‌کند.

۲. مقدار x از مکان حافظه پایدار را در C کپی می‌کند.

۳. فیلد dirty مربوط به C را پاک می‌کند (C را به‌عنوان بروز رسانی شده اعلام می‌کند).

۴. ۴. فهرست حافظه نهان را به‌روز می‌کند تا مشخص نماید که اکنون مقدار x در C ذخیره‌شده است.

اگر در گام (۱) اسلات C اشغال‌شده بود، می‌گوییم که x را جایگزین قلم داده‌ایی که در C ذخیره‌شده است، می‌کند. یعنی، برای جایگزینی اسلات، از الگوریتم‌های مشهور سیستم‌عامل نظیر الگوریتم‌های LRU و FIFO استفاده می‌شود.

مدیر حافظه نهان، برای خواندن x، چنانچه مقدار x قبلاً در حافظه نهان وجود نداشته باشد، مقدار x را به حافظ نهان واکشی می‌کند و این مقدار را از حافظه نهان برمی‌گرداند. مدیر حافظه نهان، برای نوشتن، اگر x قبلاً نوشته‌شده نباشد، ابتدا یک اسلات خالی برای x تخصیص می‌دهد، مقدار جدید را در اسلات حافظه نهان می‌نویسد، و فیلدهای dirty را برای اسلات حافظه نهان فعال می‌کند. سپس، بسته به مدیر الگوریتم ترمیم، مقدار جدید را در این مقطع زمانی یا بعداً در حافظه پایدار ثبت می‌کند.

مواردی وجود دارد که مدیر ترمیم باید مطمئن شود که یک اسلات در حافظه نهان برای مدتی (به‌عنوان‌مثال تا زمانی که محتوی اسلات در حال تغییر است) منتقل نگردد. به همین دلیل، مدیر ترمیم دو عمل اضافی Pin و Unpin را انجام می‌دهد. عمل Pin(C) به مدیر حافظه نهان می‌گوید که تا زمانی که اسلات UnPin(C) نشده است را به حافظه پایدار منتقل نکند. بنابراین، مدیر ترمیم هرگز وقتی‌که اسلاتی Pin شده باشد را در حافظه پایدار نمی‌نویسد.

۱۱-۵. الگوریتم REDO/ UNDO

در این بخش یک الگوریتم مدیر ترمیم که هم به UNDO و هم به REDO نیاز دارد را شرح می‌دهیم. این الگوریتم پیچیده‌ترین نوع الگوریتم در ۴ نوع بیان‌شده است. هرچند، مزیت اصلی‌اش نیز انعطاف‌پذیری خیلی

زیاد در انتقال اسلات‌های تغییریافته (کثیف) حافظه نهان در پایگاه داده پایدار است. این الگوریتم تصمیم‌گیری در مورد Flush را تقریباً به‌طورکلی به مدیر حافظه نهان واگذار می‌کند. این کار به دو دلیل قابل توصیف است:

۱. مدیر ترمیمی که از REDO / UNDO استفاده می‌کند از مجبور نمودن مدیر حافظه نهان به Flush غیرضروری اجتناب می‌کند، بنابراین، عملیات ورودی / خروجی (I/O) حداقل می‌شود. در نقطه مقابل، یک الگوریتمی که قابلیت REDO ندارد، بیش‌تر Flush می‌کند، چون قبل از این‌که تراکنش‌ها تثبیت شوند باید مطمئن شود تا اثرات تراکنش‌ها در پایگاه داده پایدار به‌روز شده‌اند.

۲. این الگوریتم به مدیر ترمیم اجازه می‌دهد تغییر درجا را برای جایگزین کردن اسلات تغییر کرده آخرین نوشتن توسط یک تراکنش تثبیت‌شده استفاده کند. درصورتی‌که یک الگوریتم فاقد قابلیت UNDO نمی‌تواند در این حالت اسلات را Flush کند، چون با انجام این عمل حاصل یک تراکنش تثبیت نشده را در پایگاه داده پایدار به‌روز می‌کند. به‌طورکلی، الگوریتم REDO/UNDO حداکثر کارایی را برای عمل معمولی دارد، در زمان وقوع خرابی تاثیر بیش‌تری نسبت به سایر الگوریتم‌ها دارد.

فرض کنید تراکنش T_i مقدار v را در قلم داده x می‌نویسد. در این الگوریتم، مدیر ترمیم درصورتی‌که x در حافظه نهان وجود نداشته باشد، آن را واکشی می‌کند، سپس v را در کارنامه در اسلات حافظه نهان x (یعنی C) تثبیت می‌کند، اما از مدیر حافظه نهان درخواست نمی‌کند تا C را Flush کند. مدیر حافظه نهان فقط زمانی که می‌خواهد C را برای واکشی دیگر x آزاد کند، Flush می‌کند. بنابراین، به‌روزرسانی دست مدیر حافظه نهان است نه دست مدیر ترمیم. اگر مدیر حافظه نهان C را جایگزین کند یا T_i سقط می‌شود یا خرابی سیستم قبل از تثبیت T_i پیش می‌آید،آنگاه UNDO نیاز می‌باشد. از طرف دیگر، اگر T_i تثبیت شود و قبل از این‌که مدیر حافظه نهان C را جایگزین کند، سیستم خراب گردد، آنگاه REDO نیاز می‌باشد.

فرض می‌کنیم که واحدهای اقلام داده که رویه‌های مدیر ترمیم روی آن‌ها عمل می‌نماید همان واحد انتقالی اتمیک به حافظه پایدار است. فرض می‌کنیم یک کارنامه فیزیکی که یک لیست مرتب‌شده‌ایی از رکوردهای $[T_i, x, v]$ است و بر اساس قانون زباله روبی، ترمیم می‌شود و مقدار اولیه هر قلم داده قبل از این‌که تراکنشی پردازش گردد، در کارنامه نوشته می‌شود. هر به‌روزرسانی کارنامه و لیست تراکنش‌های تثبیت‌شده قبل از ادامه پردازش مستقیماً روی حافظه پایدار ذخیره می‌گردد و تأییدیه آن نیز گرفته می‌شود.

اکنون شرح کلی پنج رویه مدیر ترمیم (RM) این الگوریتم عبارت‌اند از که در شکل ۴-۵ به طور خلاصه آمده است:

1. **RM-Write(T_i, x, v)**
   - اگر T_i قبلاً در لیست تراکنش فعال قرار نگرفت، آن را به لیست اضافه می‌کند.
   - اگر x در حافظه نهان وجود ندارد، آن را واکشی می‌کند.
   - رکورد $[T_i, x, v]$ را به کارنامه اضافه می‌کند.
   - v را در اسلات x در حافظه نهان می‌نویسد.
   - تأییدیه پردازش RM-Write(T_i, x, v) را به زمان‌بند اعلام می‌نماید.
2. **RM-Read(T_i, x)**
   - اگر x در حافظه نهان وجود ندارد، آن را واکشی می‌کند.

- مقدار اسلات x در حافظه نهان را به زمان‌بند برمی‌گرداند.

3. **RM-Commit(T_i)**

- T_i را به لیست تراکنش‌های تثبیت‌شده اضافه می‌کند.
- تأییدیه تثبیت شدن T_i را به زمان‌بند اعلام می‌کند.
- T_i را از لیست تراکنش‌های فعال حذف می‌کند.

4. **RM-Abort(T_i)**

- برای هر قلم داده‌ای x که توسط T_i به‌روز شده است:
  - اگر x در حافظه نهان وجود نداشته باشد، یک اسلات به آن تخصیص می‌دهد.
  - تصویر قبل x نسبت به T_i را در اسلات حافظه نهان کپی می‌کند.
- T_i را به لیست تراکنش‌های سقط شده اضافه می‌کند.
- تأییدیه سقط‌شدن T_i را به زمان‌بند اعلام می‌کند.
- T_i را از لیست تراکنش‌های فعال حذف می‌کند.

5. **Restart**

- همه اسلات‌های حافظه نهان را دور می‌ریزد.
- Redone:={} و Undone:={} قرار می‌دهد.
- از آخرین رکورد کارنامه به ابتدای فایل پیمایش می‌کند. گام‌های زیر را تکرار می‌کند تا این‌که یا Redone و Undone مساوی مجموعه همه اقلام داده‌ی در پایگاه داده شود، یا این‌که هیچ رکوردی در کارنامه وجود نداشته باشد. به ازای هر ورودی $[T_i, x, v]$، اگر $x \notin \text{Redone} \cup \text{Undone}$، آنگاه:

۱. اگر x در حافظه نهان وجود نداشته باشد، اسلات برای x تخصیص می‌دهد.

۲. اگر T_i در لیست تراکنش‌های تثبیت‌شده باشد، x را در اسلات x حافظه نهان کپی می‌کند و x را به مجموعه Redone اضافه می‌کند (یعنی $\text{Redone} := \text{Redone} \cup \{x\}$).

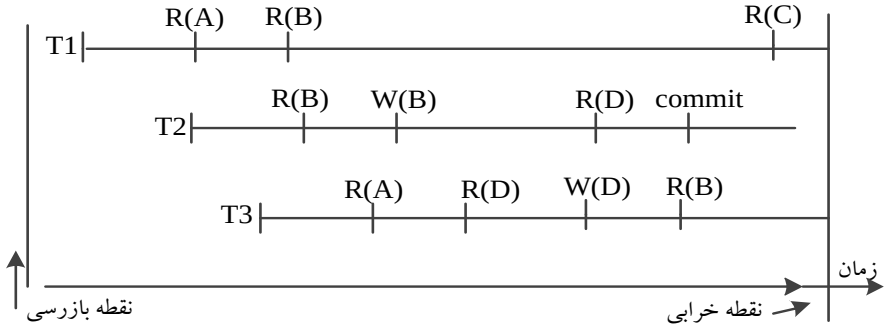
شکل۴-۵. نمایش فرآیند الگوریتم ترمیم به طور خلاصه.

در غیر این صورت (یعنی، T_i در لیست تراکنش‌های سقط‌شده یا فعال است، اما در لیست تراکنش‌های تثبیت‌شده نیست)، تصویر قبل x نسبت به نوشتن T_i را در اسلات حافظه نهان مربوط به x کپی می‌کند و x را به مجموعه Undone اضافه می‌کند (یعنی Undone := Undone ∪ {x}).

- برای هر تراکنش T_i در لیست تراکنش‌های تثبیت‌شده، اگر T_i در لیست تراکنش‌های فعال باشد، آن را از آنجا حذف می‌کند.
- تأییدیه کامل شدن Restart را به زمان‌بند اعلام می‌کند.

۲۲-۵. مسائل حل‌شده

مثال ۱. اطلاعات کارنامه با سه تراکنش به‌صورت زیر را در نظر بگیرید. فرض کنید که پروتکل به‌روزرسانی درجا (بلافاصله یا فوری) با نقطه بازرسی برای ترمیم خرابی استفاده می‌شود. دقت کنید که فایل کارنامه لزوماً تمام عملیات یک تراکنش را ثبت نمی‌کند.



آیا این زمان‌بند ترمیم‌پذیر است؟

این زمان‌بند ترمیم‌پذیر نیست، زیرا، تراکنش T_2، مقدار D که مقدار آن قبلا توسط تراکنش T_3 نوشته‌شده است را می‌خواند، اما T_2 قبل از تثبیت T_3، تثبیت شده است.

مثال ۲. فرض کنید که کارنامه زیر شامل عملیات تراکنش‌ها از شروع اجرای سیستم پایگاه داده باشد که از کارنامه گیری Undo/Redo به همراه نقطه بازرسی برای ترمیم استفاده کند:

1) <START T_1>
2) < T_1, X, 20, 10>
3) < T_1, Y, 40, 0>
4) <START T_2>
5) < T_1, X, 50, 20>
6) < T_2, Z, 30, 20>
7) <COMMIT T_1>
8) <START T_3>
9) < T_3, U, 60, 30>
10) < T_2, V, 50, 25>
11) <START CKPT (T_2, T_3)>
12) < T_2, Z, 45, 30>
13) <COMMIT T_2>
14) <START T_4>
15) <T_4, W, 80, 10>
16) <COMMIT T_3>
17) <END CKPT>
18) <T_4, W, 100, 80>
19) <COMMIT T_4>

همچنین ورودی‌های کارنامه برای به‌روزرسانی پایگاه داده با فرمت زیر می‌باشد:
(TransactionId, variable, newValue, oldValue)

مقدار اقلام داده x, y, z, u, v, w روی دیسک بعد از پایان ترمیم در حالات زیر چیست؟

۱. اگر سیستم دقیقا قبل از این که خط ۱۳ که بر روی دیسک نوشته شود، از کار بی‌افتد.

۲. اگر سیستم دقیقا قبل از این که خط ۱۴ که بر روی دیسک نوشته شود، از کار بی‌افتد.

۳. اگر سیستم دقیقا قبل از این که خط ۱۹ که بر روی دیسک نوشته شود، از کار بی‌افتد.

پاسخ:

1. x = 50, y = 40, z = 20, u = 30, v = 25, w =10
2. x = 50, y = 40, z = 45, u = 30, v = 50, w =10
3. x = 50, y = 40, z = 45, u = 60, v = 50, w =10

مثال ۳. محتویات کارنامه خنثی (Undo log) را به‌صورت زیر در نظر بگیرید:

LSN1 <START T_1>
LSN2 < T_1 × 5>

LSN3 <START T_2>
LSN4 < T_1 × 7>
LSN5 < T_2 × 9>
LSN5 <START T_3>
LSN6 < T_3 z 11>
LSN7 <COMMIT T_1>
LSN8 <START CKPT(T_2, T_3)>
LSN9 <START T_1>
LSN10 < T_2 × 13>
LSN11 < T_3 × 15>
*C*R*A*S*H*

الف: نشان دهید که مدیریت ترمیم چه تعداد عقب‌گرد برای خواندن از کارنامه نیاز دارد. اولین LSN که مدیر ترمیم می‌خواند را بنویسید.

ب: عملیات ترمیم را نشان دهید؟

ج: مقدار x در پایان ترمیم چند است؟

الف: LAN3 ب: X = 15، X = 13، Z = 11 ، X = 9 ج: X = 9

۲۳-۵. سؤالات چهارگزینه‌ای

۱. با به وجود آمدن یک نقطه تثبیت (Commit Point)، کدام مورد برقرار نیست؟

الف: تمام به هنگام سازی‌ها از حالت موقت خارج و پایدار می‌شوند.

ب: تمام آدرس‌دهی‌های بانک اطلاعاتی از بین می‌رود و قفل چندتایی‌ها باز می‌شود.

ج: قبل از رسیدن به نقطه تثبیت، رکورد Checkpoint در فایل ثبت نوشته می‌شود.

د: ممکن است به هنگام سازی‌ها هنوز در حافظه اصلی و بافر موجود باشند.

۲. اگر تراکنشی بعد از نقطه وارسی (نقطه بازرسی Checkpoint) عمل تثبیت (Commit) داشته باشد، در صورت خرابی سیستم کدام‌یک از کارهای زیر انجام می‌گیرد.

الف: Undo انجام می‌شود. ب: Redo انجام می‌شود.

ج: کار خاصی انجام نمی‌گیرد. د: Rollback انجام می‌شود.

۳. کدام واحد از سیستم‌های مدیریت پایگاه داده‌ها وظیفه ترمیم خرابی را بر عهده دارد؟

الف: مدیر ترمیم ب: مدیر فایل ج: مدیر صفحه د: مدیر دیسک

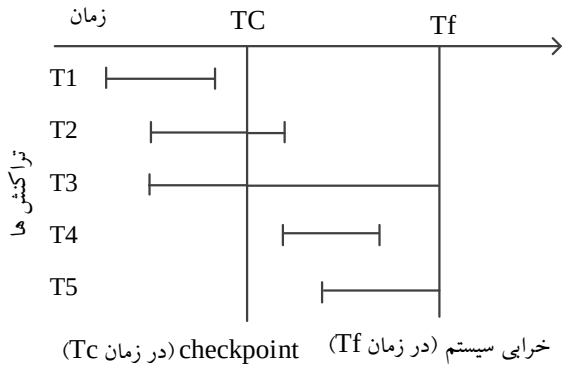
٤. درصورتی‌که پایگاه داده دچار آسیب و خرابی گسترده‌ای شود با استفاده از کدام روش می‌توان آن را ترمیم کرد؟

الف: استفاده از نسخه پشتیبان و انجام عمل Redo با کمک فایل ثبت سوابق

ب: استفاده از نسخه پشتیبان

ج: با انجام عمل Redo با کمک فایل ثبت سوابق د: با انجام عمل Redo

۵. با توجه به شکل زیر گزینه درست را انتخاب کنید.



الف: تراکنش T1 باید دوباره انجام شود. ب: تراکنش‌های T3 و T5 باید لغو شوند.

ج: تراکنش‌های T2 و T4 نیاز به اجرای دوباره نیستند. د: تراکنش T1 و T2 باید دوباره انجام شوند.

۶. فایل ثبت (کارنامه log) در واحد پایگاه داده چیست؟

الف: در این فایل داده‌های پایگاه ثبت می‌شود.

ب: در این فایل هم داده‌ها و هم چگونگی اجرای تراکنش، ثبت می‌گردد.

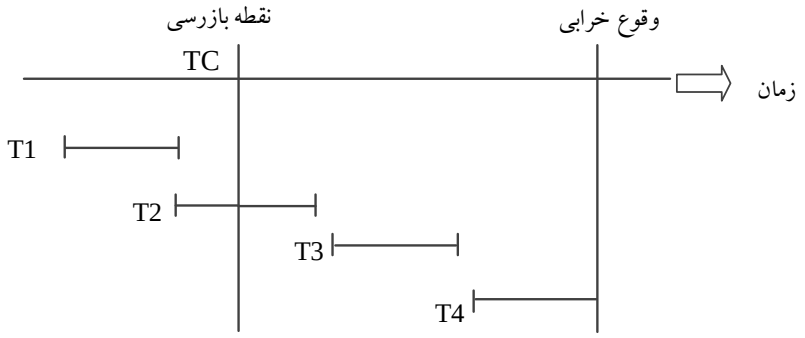
ج: در این فایل هم چگونگی اجرا تراکنش و هم عملیاتی که روی یک فقره داده انجام می‌شود، ثبت می‌گردد.

د: فایل ثبت در واحد ترمیم وجود ندارد.

۲۴-۵. پاسخ سؤالات چهارگزینه‌ای

۱. **گزینه (ج) صحیح است.** چون اگر نقطه تثبیت ایجاد شود، یک واحد کار منطقی با موفقیت به پایان می‌رسد. یعنی، تمام به هنگام رسانی های پایگاه داده که توسط تراکنش در حال اجرا بعد از نقطه تثبیت قبلی، انجام‌شده‌اند، تثبیت می‌شوند (لذا گزینه (الف) صحیح است)، گزینه دیگری که در هنگام نقطه تثبیت اتفاق می‌افتد، تمام آدرس‌دهی (مثلاً از طریق اشاره‌گرها در SQL) از بین خواهند رفت و قفل تاپل‌ها (چندتایی‌ها) آزاد می‌شوند. دقت کنید، که در نقطه تثبیت یک واحد کاری به اتمام می‌رسد، تراکنش‌های دیگر که به‌طور موازی با این تراکنش اجرا می‌شوند که ممکن است تغییراتی در قسمت‌های خودشان نشان بدهند. پس با اطمینان نمی‌توان بیان کرد که کل پایگاه داده در نقطه تثبیت در وضعیت صحیح قرار می‌گیرد، بنابراین گزینه (د) نیز درست می‌باشد.

۲. **گزینه (ب) صحیح است.** با توجه به شکل زیر، تراکنش‌های T_2 و T_3 به دلیل رعایت خاصیت پایایی یا ماندگاری تراکنش باید تکرار شوند.



۳. **گزینه (الف) صحیح است.** چون واحد مدیریت ترمیم (Recovery Management) وظیفه جلوگیری از تأثیر تراکنش‌های نیمه‌کاره بر روی بانک اطلاعات و از بین بردن آثار آن‌ها را به عهده دارد.

۴. **گزینه (الف) صحیح است.** چون اگر رسانه (حافظه جانبی) آسیب ببیند، به‌طوری‌که اطلاعات آن قابل بازیابی نباشد، برای ترمیم ابتدا نسخه پشتیبان را بازیابی کرده، سپس دستوراتی که در مدت‌زمان پشتیبان گیری تا خرابی تثبیت‌شده‌اند را به کمک فایل کارنامه (Log) مجدداً اجرا (REDO) می‌کنیم.

فصل ۶
امنیت در پایگاه داده

۵. **گزینه (ب) صحیح است.** تراکنش‌های T_2 و T_4 باید REDO (دوباره اجرا) شوند. چون بعد از آخرین نقطه بازرسی (Checkpointing) تا زمان خرابی تثبیت (Commit) شده‌اند. اما تراکنش‌های T_3 و T_5 باید لغو (خنثی یا UNDO) شوند، زیرا، پس از آخرین نقطه بازرسی تا زمان خرابی تثبیت نشده‌اند.

اکثر افراد قبل از ارسال نامه، آن را در پاکت قرار می‌دهند. اگر بپرسیم چرا این کار را انجام می‌دهید، برخی پاسخ‌های زیر را می‌شنویم:

"واقعاً نمی‌دانم"، "از روی عادت"، "زیرا، بقیه این کار را انجام می‌دهند".

آیا این پاسخ‌ها واقعاً صحیح هستند؟ یا دلایل دیگری وجود دارد. افراد در اصل برای جلوگیری از خواندن نامه‌ها توسط دیگران آن را درون پاکت قرار می‌دهند. حتی، اگر محتوی نامه شامل اطلاعات مهم یا شخصی نباشد، بسیاری از افراد برای این‌که مطمئن شوند، مکاتبات شخصی آن‌ها خصوصی می‌ماند آن را در پاکت قرار می‌دهند و مهر می‌کنند تا از دید افراد دیگر مخفی بماند و به دست گیرنده برسد. چون، اگر نامه را در یک پاکت درباز قرار دهند و آن را ارسال نمایند، در بین راه افراد به‌راحتی می‌توانند محتوی آن را بخوانند و تغییر دهند، به‌طوری‌که راهی برای تشخیص خواندن و تعویض آن وجود ندارد.

امروزه، اکثر افراد از پست الکترونیک[۱۴] به‌جای ارسال نامه‌ها از طریق اداره پست استفاده می‌کنند. پست الکترونیک ابزاری سریع برای انتقال‌نامه است. اما، در آن پاکتی برای محافظت از محتوی نامه (اطلاعاتی که از طریق پست الکترونیک ارسال می‌شود)، وجود ندارد. درواقع، ارسال نامه از طریق پست الکترونیک شبیه به پست نمودن نامه بدون پاکت است. بنابراین، اگر فردی بخواهد پیام شخصی یا اطلاعات محرمانه را از طریق پست الکترونیک ارسال کند، باید راهی جهت محافظت از نامه خود (جلوگیری از خواندن و تغییر توسط افراد دیگر) پیدا کند.

رایج‌ترین راه‌حل، استفاده از پنهان‌سازی است. زیرا، این روش پیام را رمز می‌کند. در این حالت، اگر پیام رمز شده (کد شده) به دست فرد دیگری (به‌جز گیرنده) برسد، آن فرد با خواندن پیام چیزی از اطلاعات آن نمی‌فهمد.

۱- ۶. اهمیت امنیت اطلاعات

قبل از این‌که به اهمیت امنیت اطلاعات بپردازیم، مقوله امنیت را درگذشته موردبحث قرار می‌دهیم. همان‌طور که می‌دانید، امنیت در زمان‌های ماقبل تاریخ نیز جایگاه ویژه‌ای داشته است. در آن زمان انسان‌ها از جان خود در مقابل حیوانات وحشی محافظت می‌کردند. ابتدا، انسان‌ها برای خودشان خانه‌هایی ساختند تا از گزند حیوانات وحشی در امان بمانند. سپس، برای درب‌های خانه قفل‌هایی را تعبیه کردند یا نگهبان استخدام نمودند تا از اموال خانه (سرمایه) محافظت نمایند. با توجه به میزان اهمیت چیزی که از آن محافظت می‌شود، امنیت می‌تواند لایه‌ها و سطوح مختلف داشته باشد. به‌عنوان‌مثال، یک طلافروشی را در نظر بگیرید. طلافروش برای برقراری امنیت سه لایه (سطح) امنیتی را ایجاد می‌کند که عبارت‌اند از:

۱.درب مغازه طلافروشی را می‌بندد.

۲.طلاها را در گاوصندوق ضد سرقت قرار می‌دهد.

۳.طلافروشی را به سیستم ضد سرقت مجهز می‌نماید.

حال، یک نانوایی را در نظر بگیرید. نانوا، پس از خاتمه کار نانوایی، فقط درب نانوایی را می‌بندد (یعنی نیازی به گاوصندوق و سیستم ضد سرقت ندارد). بنابراین، همان‌طور، که بیان گردید، سطوح امنیتی که برای طلافروشی ایجاد می‌کنیم، خیلی قوی‌تر و پیچیده‌تر از یک نانوایی است. زیرا، ارزش یک کیلوگرم طلا به‌مراتب بیشتر از چند کیسه آرد می‌باشد. پس ، امنیت برای چیزهایی مطرح می‌شود که مهم هستند و ارزش زیادی دارند. یعنی، برقراری امنیت برای سرمایه‌هایی از قبیل پول، طلا، عکس‌ها و فیلم‌های خانوادگی، رازهای زندگی، رمز کارت‌های حساب بانکی و غیره مهم هستند.

---

[14].Email [2].Cryptography

اکنون این سؤال مطرح می‌شود، آیا اطلاعات در فضای کامپیوتر (مجازی) سرمایه هستند یا خیر؟ آیا این اطلاعات نیاز به محافظت دارند یا نه؟

برای این‌که به این سؤال پاسخ دهیم، به مثال‌های زیر می‌پردازیم:

۱. فرض کنید کارمند شهرداری یکی از شهرهای بزرگ هستید. در سیستم رایانه شهرداری اطلاعاتی از قبیل طرح‌های نوسازی شهر، اتوبان‌هایی که قرار است در شهر ایجاد شوند و مکان آن‌ها، مکان فضاهای سبز، پارک‌ها و غیره ذخیره‌شده است. آیا این اطلاعات نیاز به محافظت دارند. در نگاه اول ممکن است فکر کنید، این اطلاعات ارزش چندانی ندارند و نیاز به محافظت ندارند. اما، اگر این اطلاعات به دست افراد خاصی برسند، می‌توانند درآمدهای چندمیلیاردی از آن کسب کنند. زیرا، این افراد زمین‌های اطراف این مکان‌ها را به قیمت خیلی پایین می‌خرند و پس از مدت کوتاهی این زمین‌ها را با چندین برابر قیمتی که خریداری کردند، می‌فروشند.

۲. فرض کنید در شرکتی کار می‌کنید که در مناقصات میلیاردی شرکت می‌کند و اطلاعات پیشنهادی قیمتش را در رایانه شرکت ذخیره می‌کند. از طرف دیگر، فرض کنید، فقط چند شرکت در این مناقصات شرکت می‌کنند. اگر یک شرکت بتواند اطلاعات پیشنهاد قیمت شرکت‌های دیگر را به دست آورد، به‌راحتی می‌تواند در مناقصه برنده شود.

۳. شرکتی را در نظر بگیرید که مواد اولیه کارخانه‌های کشور را خریداری می‌کند. این شرکت مواد اولیه موردنیاز را در فایل‌های اکسل و **Word** ذخیره می‌کند و بر روی رایانه شرکت نگه‌داری می‌نماید. آیا این اطلاعات نیاز به محافظت دارند؟ برای پاسخ به این سؤال، سؤال دیگری مطرح می‌شود. اگر این اطلاعات در اختیار دشمنان کشورمان قرار بگیرند، چه مشکلی را ایجاد می‌کند؟ چون این مواد از کشور خارجی خریداری می‌گردد و چنانچه مشخص باشد، مواد اولیه از کدام کشور خریداری می‌شود، دشمنان با تنگ کردن حلقه‌ی تحریم، موجب جلوگیری از فروش آن مواد به ایران می‌شوند.

٤. فرض کنید، کارتی دارید که از طریق آن از عابر بانک‌ها پول برداشت می‌کنید. آیا کارت و کلمه عبور آن را در اختیار غریبه قرار می‌دهید؟

۵. فرض کنید، از طریق اینترنت خریدوفروش الکترونیکی انجام می‌دهید. اگر بدانید که فضای اینترنت امن نیست و ممکن است سرمایه‌تان سرقت شود، آیا در این فضا خریدوفروش انجام خواهید داد؟

از طرف دیگر، شاید در خبرها شنیده باشید که بی‌توجهی و سهل‌انگاری در برقراری امنیت اطلاعات، خسارت‌های زیادی را به افراد حقیقی و حقوقی وارد کرده است. چند نمونه از این خبرها در زیر آمده‌اند:

۱. یک هکر هزینه‌ای برابر با دوازده هزار دلار را روی دست آژانس فدرال مدیریت آژانس آمریکا گذاشت.

۲. گروهی از هکرها ادعا می‌کنند که توانسته‌اند به صندوق پست الکترونیکی خانم سارا پلین (معاون نامزد جمهوری خواهان در انتخاب ریاست جمهوری ایالات‌متحده) دست یابند.

۳. حساب بانکی رئیس‌جمهور فرانسه (نیکولا سارکوزی) هک شد.

٤. یک هکر که به حساب‌های بانکی شهروندان تهرانی نفوذ می‌کرد، به دام افتاد.

۵. به فاصله کوتاهی پس از عبور تانک‌های ارتش روسیه از مرزهای گرجستان، وب‌سایت دولتی گرجستان مورد آماج حمله قرار گرفت.

٦. بنا بر تخمین شرکت سمانتیک امروزه ۱/۲ درصد از پیام‌های پست الکترونیک حاوی بدافزارهای مخرب هستند.

۷. به‌تازگی هرزنامه جدیدی در فضای اینترنت منتشرشده است که وانمود می‌کند، از سوی جان پیستول، معاون مدیر FBI صادرشده است.

۸. یک رایانه دست‌دوم که اطلاعات کارت اعتباری تعدادی از مشتریان بانک انگلیسی بر روی آن ذخیره‌شده بود، در جریان یک بی‌احتیاطی فروخته شد.

۹. تعداد کل بدافزارهای تولیدشده در سال ۲۰۰۷ برابر کل بدافزارهای تولیدشده در ۱۵ سال قبل آن بوده است.

۱۰. بنا بر اعلام شرکت سمانتیک ۷۸ درصد از حجم کل پست‌های الکترونیکی جهان از اسپم تشکیل‌شده است.

۱۱. آیا موساد پشت نرم‌افزارهای اسرائیلی پنهان‌شده است؟

پیدا کردن برخی اطلاعات شخصی و پاسخ به سؤالات امنیتی می‌تواند از طریق جست‌وجو در اینترنت به‌راحتی صورت گیرد. یکی از هکرها به همین صورت موفق گردید، به پست الکترونیکی خانم سارا پلین دسترسی پیدا کند.

علاوه بر این خبرها، هر روز هزاران خبر دیگر را درزمینهٔ سرقت اطلاعات در اینترنت می‌بینید. از طرف دیگر، امروزه با توسعه رایانه‌ها و گسترش شبکه‌های رایانه‌ای مانند اینترنت، امنیت شبکه‌های رایانه‌ای و اطلاعات به امری بسیار مهم و حیاتی تبدیل گردیده است.

۲ – ۶. مفاهیم اولیه امنیت اطلاعات

قبل از این‌که به امنیت اطلاعات و روش‌های برقراری آن بپردازیم، با چند مفهوم در امنیت آشنا می‌شویم. برخی از این مفاهیم در زیر آمده‌اند:

امنیت[۱۵]، به حراست و حفاظت از منابع اطلاعات امنیت گویند. امنیت می‌تواند توسط هر شخصی یا برنامه‌ای نقص شود. اگر نتیجه فاجعه‌آمیزی روی کاربران یا محیط پیش نیاید، می‌گویند سیستم امن خواهد بود.

تهدید[۲]، پدیده یا رویدادی که بتواند امنیت سیستم را به خطر بی‌اندازد. برخی از این پدیده‌ها و رویدادها سیل، زلزله، آتش‌سوزی، طوفان، تصادف و غیره می‌باشند.

حمله[۳]، تلاش آگاهانه و حساب‌شده‌ای است تا بتوان نقاط ضعف سرویس‌های امنیتی را شناخته و از نقص سیستم‌های امنیتی استفاده کرده، سیستم را موردحمله قرارداد تا به آن آسیب رسانده یا از داده‌های آن استفاده کرد.

آسیب‌پذیری[۴]، ضعف‌های موجود در نرم‌افزار، سیستم یا دیگر مکانیزم‌هایی که یک رخنه‌گر برای دسترسی به سیستم یا شبکه می‌تواند از آن بهره بگیرد.

ریسک[۵]، احتمال این‌که ضعف‌ها یا آسیب‌پذیری سیستم شناخته‌شده و از آن‌ها استفاده گردد.

افشا[۶]، هزینه اتلاف (خسارت) تخمینی حاصل از سوءاستفاده یک تهدید از یک آسیب‌پذیری را گویند. افشا وقتی به وجود می‌آید که سیستم رایانه‌ای:

۱. اجازه می‌دهد مهاجم فعالیت‌های جمع‌آوری اطلاعات را انجام دهد.

۲. اجازه می‌دهد مهاجم فعالیت‌هایی را مخفی نماید.

۳. دارای قابلیتی است که رفتار مورد انتظار را انجام می‌دهد. ولی، به‌آسانی درخطر کشف قرار می‌گیرد.

٤. اولین نقطه ورودی است که یک مهاجم ممکن است برای دسترسی به سیستم یا داده استفاده کند.

شبکه‌های بات[۱۶]، تعداد زیادی (مثلاً صدها هزار) رایانه اینترنتی سرقت شده که از آن‌ها برای هدایت ترافیک شامل هرزنامه و ویروس به دیگر رایانه‌های اینترنتی استفاده می‌شوند.

حمله‌کننده[۲] **یا هکر**[۳]، فردی است که همواره درصدد استفاده از نقاط ضعف و آسیب‌پذیری موجود در سیستم می‌باشد. این افراد هدف غیر مخرب ندارند و حاصل عملیات آن‌ها می‌تواند اثرات جانبی منفی را به دنبال داشته باشد.

نفوذ گران، افرادی هستند که بدون مجوز از منابع سیستم استفاده می‌کنند.

بدافزار[۴] یا **کد مخرب**، شامل ویروس‌ها، کرم‌ها و برنامه‌های تروجان بوده که هر یک از آنان دارای ویژگی‌های منحصربه‌فردی هستند.

---

[1].Security [2].Threat [3]. Attack [4]. Vulnerability [5]. Risk
[6].Exposure
[16].BotNet [2] .Attacker [3].Hacker [4].MalWare [5].Trojan

ویروس‌ها، نوع خاصی از کد مخرب هستند که برای آلوده کردن سیستم، عملیات خاصی را انجام می‌دهند. این نوع برنامه‌ها، برای نیل به اهداف تخریبی خود نیاز به یاری کاربران دارند. نمونه‌هایی از همکاری کاربران عبارت‌اند از:

1. اجرای یک برنامه آلوده در سیستم.
2. باز کردن یک فایل پیوست آلوده با پست الکترونیک.
3. مشاهده یک وب‌سایت آلوده خاص و غیره.

کرم، کد مخربی است که بدون نیاز به دخالت کاربر، توزیع و گسترش می‌یابد. کرم‌ها، عموماً با سوءاستفاده از نقطه آسیب‌پذیری در سیستم یا نرم‌افزار فعالیت خود را شروع کرده و سعی می‌کنند رایانه را نیز آلوده نمایند. پس از آلوده کردن یک رایانه، تلاش می‌کنند سایر رایانه‌ها را آلوده نمایند. مانند ویروس‌ها، کرم‌ها نیز می‌توانند از طریق پست الکترونیک، وب‌سایت‌ها و نرم‌افزارهای مبتنی بر شبکه توزیع و گسترش یابند. یکی از تفاوت‌های بسیار مهم کرم‌ها و ویروس‌ها، توزیع اتوماتیک کرم‌ها می‌باشد.

برنامه‌های تروجان[5]، نرم‌افزارهایی هستند که ادعای ارائه خدمات را دارند. ولی در عمل، اهداف خاص خود (تخریب) را دنبال می‌کنند. به‌عنوان‌مثال، برنامه‌ای که ادعای افزایش سرعت رایانه‌تان را دارد، ممکن است در عمل اطلاعات مهم رایانه‌تان را برای یک مهاجم و سارق راه دور ارسال نماید.

۳ – ۶. امنیت در پایگاه داده

اکنون که با مفهوم امنیت آشنا شدیم، به امنیت در پایگاه داده می‌پردازیم. امنیت در پایگاه داده محافظت از تلاش‌های تبهکارانه جهت سرقت، تغییر و یا تخریب داده‌های پایگاه داده را تعریف می‌کند. امنیت ازجمله مفاهیمی است که باید در تمام لایه‌های یک سیستم رایانه‌ای اعم از سخت‌افزار، شبکه، سیستم‌عامل، پایگاه داده و ... آن را لحاظ نمود. تهدیدات امنیت به دو صورت عمدی و غیرعمدی هستند. دسته اول شامل تمام تهدیداتی است که نتیجه تخلف عمدی کاربران و یا برنامه‌ها است. این کاربران می‌توانند کاربران مجاز سیستم و یا کاربران خارجی (که می‌توانند به‌صورت غیرمجاز به سیستم و منابع آن دسترسی داشته باشند) باشند. برنامه‌ها نیز می‌توانند هر برنامه محلی و یا هر برنامه از راه دور (شامل ویروس‌ها یا اسب‌های تراوا و...) باشند. دسته دوم شامل تهدیداتی است که براثر ناآگاهی یا عدم دقت، کوتاهی افراد، نارسائی و کیفیت پایین برنامه‌ها رخ می‌دهد.

هنگامی‌که صحبت از یک پایگاه داده امن به میان می‌آید معمولاً سه هدف زیر در رابطه با آن مطرح می‌شود:

۱. **محرمانگی**[۱۷]: محرمانگی به صورت عدم دسترسی کاربران غیرمجاز به اطلاعات تعریف می‌شود. به عنوان مثال، یک دانشجو نباید اجازه مشاهده اطلاعات سایر دانشجویان را داشته باشد.

۲. **جامعیت**[۲]: فقط کاربران مجاز می‌توانند داده‌های مربوطه را تغییر دهند. به‌عنوان‌مثال، دانشجویان می‌توانند نمرات خود را ببینند اما اجازه تغییر آن را ندارند. جامعیت به‌صورت جلوگیری از تغییر، حذف یا دخالت ناخواسته و نادرست در اطلاعات نیز تعریف می‌شود.

۳. **دسترس‌پذیری**[۳]: مجوز کاربران مجاز نباید به‌طور ناخواسته قطع شود. به‌عنوان‌مثال، استاد درسی که اجازه تغییر نمرات را دارد، همواره باید بتواند این عمل را انجام دهد.

برای رسیدن به سه هدف فوق باید سیاست‌های امنیتی واضح و مشخصی تدوین گردد. به‌عبارت‌دیگر، باید به‌طور کامل و صریح روشن گردد که چه بخش یا بخش‌هایی از داده‌ها باید محافظت شوند و چه کاربرانی اجازه دسترسی به چه قسمت‌هایی از داده‌ها را دارند.

برای برقراری امنیت در سیستم پایگاه داده، ابتدا با کمک مکانیزه‌های تصدیق هویت کاربر مثل کلمه عبور اطمینان حاصل می‌کنیم که کاربر واردشده به سیستم یک کاربر مجاز است. البته این امکان نیز وجود دارد که کاربری غیرمجاز به نحوی بتواند مجوز ورود را به دست آورد (مثلاً از کلمه عبور یک کاربر مجاز استفاده کند). در مرحله بعد که مطمئن هستیم کاربری که وارد سیستم شده اجازه ورود داشته است، با مکانیزه‌های کنترل دسترسی (مجوز دهی به کاربر)، فقط مجوزهای دسترسی به داده‌های به خصوصی که می‌تواند به آن‌ها دسترسی داشته باشد را به او تخصیص می‌دهیم. اما این کاربر مجاز ممکن است روی همان داده‌های مجاز خود نیز تغییرات غیرمجاز انجام دهد (مثل قرار دادن عددی منفی در صفت سن). مقابله با چنین تهدیداتی به تضمین جامعیت سیستم برمی‌گردد که با قرار دادن قیود جامعیت، استفاده از مکانیزه‌هایی مثل assertion، trigger و.... از داده‌های پایگاه داده مراقبت می‌نماییم.

در پایگاه‌های داده نامتمرکز بخشی از امنیت بانک به تبادل اطلاعات و ذخیره‌سازی آن‌ها بستگی دارد. به‌عبارت‌دیگر، در یک پایگاه داده نامتمرکز، به دلیل انتقال داده‌ها در شبکه و به‌خصوص بین گره‌های مدیریت پایگاه داده، می‌بایست داده‌هایی که بین گره‌های مختلف شبکه در تبادل هستند را از دید افراد و کاربران مخفی نمود و با مکانیزه‌هایی داده‌ها را کد و رمزنگاری کرد. رمزنگاری امنیت کانال‌های ارتباطی را برقرار می‌کند. ایده اصلی رمزنگاری داده‌ها استفاده از الگوریتم‌های رمزنگاری و نیز یک کلید رمزنگاری مخصوص مدیر بانک اطلاعات است که به‌صورت امن نگاه داشته می‌شود. خروجی یک الگوریتم رمزنگاری، نسخه رمز شده داده‌های ورودی است. به همین ترتیب الگوریتم‌های رمزگشایی برای باز کردن داده‌های رمز شده و برگرداندن آن‌ها به حالت اولیه است.

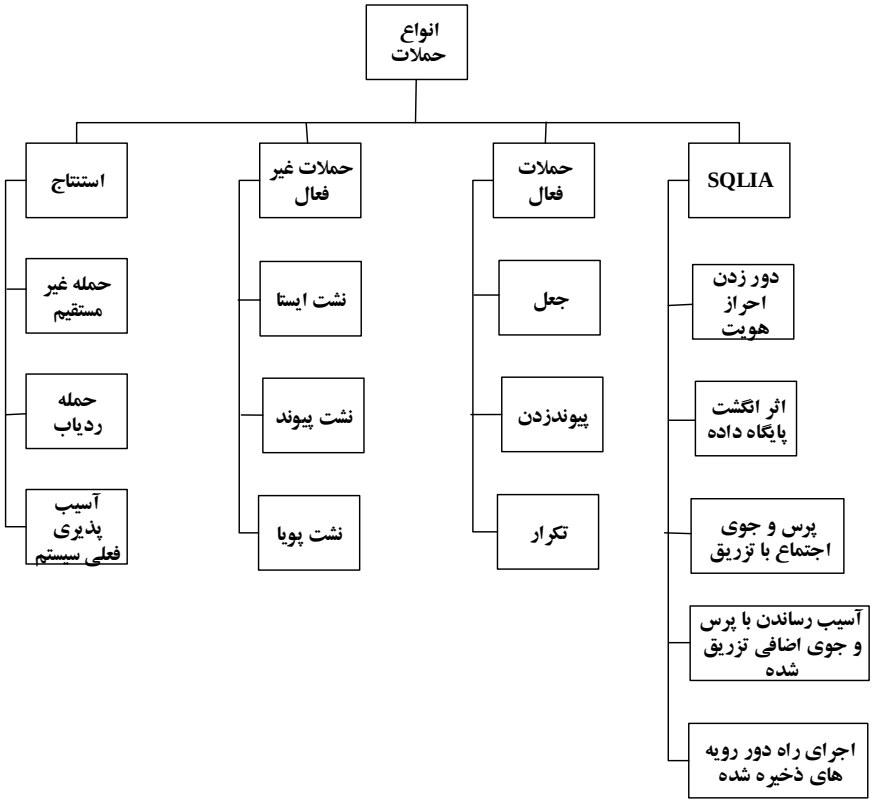
---

[17].Secrecy [2]. Integrity [3]. Availability

در پایگاه‌های داده XML و به‌خصوص بانک‌های نامتمرکز، اغلب به دلیل ذخیره‌سازی اطلاعات به فرمت رشته، عمل رمزنگاری داده‌ها جهت انتقال و همچنین ذخیره‌سازی بسیار اجباری است. در این بخش وارد بحث رمزنگاری نمی‌شویم و فقط روش‌های رمزنگاری را که به دودسته کلی متقارن و نامتقارن تقسیم می‌شوند را اجمالاً بیان می‌کنیم و در ادامه رمزگذاری پایگاه داده را بحث خواهیم کرد.

روش رمزنگاری متقارن: درروش رمزنگاری متقارن، فرستنده و گیرنده از یک کلید سری مشترک برای رمزنگاری و رمزگشایی استفاده می‌کنند. روش استاندارد رمزگذاری داده Data Encryption Standard (DES) مثالی از رمزنگاری متقارن است. بدیهی است که برای دو طرف ناشناس توافق بر روی یک کلید سری مشترک دشوار است. بنابراین، این روش مورد استفاده کم‌تری دارد.

روش رمزنگاری نامتقارن: روش دیگر، روش رمزنگاری نامتقارن است. در این روش هر فرد دو کلید در اختیار دارد، کلید عمومی که آزادانه منتشر می‌شود و کلید خصوصی که به‌صورت خصوصی و کاملاً محرمانه نگه‌داری می‌شود. در این روش، فرستنده داده‌ها را با کلید عمومی رمز کرده و به گیرنده ارسال می‌کند. داده‌های رمز شده تنها با کلید خصوصی گیرنده قابل رمزگشایی هستند و ازآنجایی‌که کلید خصوصی هر شخص منحصربه‌فرد است و به‌طور کاملاً امن نگه‌داشته می‌شود، شخص دیگری نمی‌تواند این اطلاعات را رمزگشایی کرده و یا از آن‌ها استفاده نماید. این روش از امنیت بالایی برخوردار است و مورداستفاده فراوانی دارد. به‌عنوان نمونه، الگوریتم‌های معروف RS2 ، MD4 و MD5 از این روش استفاده می‌کنند.



۶-۱ انواع حملات پایگاه داده.

۶-۱۰. سؤالات چهارگزینه‌ای

۱. محدودیت‌های امنیتی (Security constraints) بانک اطلاعاتی در کدام قسمت ذخیره می‌شود؟

الف: پرونده log ب: ردپا (Aduit Trail) ج: فرهنگ داده (Catalog) د: پرونده‌های سیستم‌عامل

۲. کدام‌یک از روش‌های زیر برای جلوگیری از سرقت اطلاعاتی استفاده می‌شود؟

الف: تعریف View ب: به رمز درآوردن اطلاعات (Encryption)

ج: کنترل اجازه دسترسی اجباری (Mandatory)

د: کنترل اجازه دسترسی احتیاط‌آمیز (Discretionary)

۳. اگر کاربری که اجازه دسترسی به رابطه پایه P را ندارد. دستور بازیابی اطلاعات از P را بدهد، کدام‌یک از پاسخ‌های زیر ازنظر امنیتی بهتر است؟

الف: No such relation exists ب: Consult system Administrator

ج: You are not allowed to access P

د: Relation P does not exist or your are not allowed to access it

٤. کدام‌یک از موارد زیر باعث رواج کنترل اجباری (Mandatory) شده است؟

الف: قابلیت اطمینان بیش‌تر ب: حمایت وزارت دفاع آمریکا

ج: تناسب بهتر با سیستم‌عامل د: تبدیل پرس‌وجو به شکل داخلی

۵. کدام عبارت در مورد یکپارچگی (Integrity) و امنیت (Security) صحیح است؟

الف: برقراری یکپارچگی منجر به برقراری امنیت می‌شود.

ب: برقراری امنیت منجر به برقراری یکپارچگی می‌شود.

ج: امنیت چندلایه‌ای باعث برقراری یکپارچگی می‌شود.

د: Audit Trail برای افزایش امنیت به کار می‌رود.

۱۱-۶. پاسخ تشریحی سؤالات چهارگزینه‌ای

۱. گزینه (ج) صحیح است. کاتالوگ شامل داده‌های از قبیل جدول پایه، دیدها، محدودیت‌ها (قیود) جامعیتی، قیود امنیتی و غیره است. چون سیستم مدیریت پایگاه داده با توجه به اطلاعات فرهنگ داده عملیات کاربران را نظارت می‌کند تا تضمین کند قیود موردنظر اعمال می‌شوند.

۲. گزینه (ب) صحیح است. چون یکی از روش‌های سرقت اطلاعات، شنود آن می‌باشد. برای جلوگیری از شنود اطلاعات (استراق سمع) از رمزگذاری استفاده می‌شود.

۳. گزینه (د) صحیح است. یکی از روش‌های اعمال امنیت دست‌کاری درخواست (Request Modification) است. از مزایای این طرح عبارت‌اند از: ۱. پیاده‌سازی آن آسان است. ۲. کارآمد است. اما عیب این طرح این است که تمام محدودیت‌ها امنیتی نمی‌تواند مدیریت شود. فرض کنید که کاربری به‌هیچ‌وجه اجازه ندارد به متغیر P دسترسی داشته باشد. در این صورت، هیچ شکل تغییریافته از دستور بازیابی اصلی نباید این توهم را ایجاد کند که متغیر رابطه‌ای P وجود ندارد. در عوض لازم است پیام خطای صریحی مبنی بر << شما اجازه ندارید به این متغیر رابطه‌ای دسترسی داشته باشید>> صادر گردد. یا سیستم می‌توانست به‌دروغ بگوید <<چنین متغیر رابطه‌ای وجود ندارد>> و یا بهتر آن‌که بگوید << چنین متغیر رابطه‌ای وجود ندارد یا شما اجازه دست‌یابی به آن را ندارید>>.

۴. گزینه (ب) صحیح است. کنترل‌های دسترسی اجباری در اویل دهه ۱۹۹۰ در دنیای بانک اطلاعاتی بسیار موردتوجه قرار گرفتند، زیرا در آن زمان وزارت دفاع آمریکا (DOD) می‌خواست هر سیستمی که می‌خرد از چنین کنترل‌هایی پشتیبانی کند، همین دلیل، فروشندگان DBMS این کنترل‌ها را پیاده‌سازی کردند.

۵. گزینه (د) صحیح است. چون حسابرسی یک فایل یا پایگاه داده‌ی است که به‌صورت خودکار کارهای انجام‌شده به‌وسیله کاربر آن ثبت می‌شود تا بتواند کاربرانی که عملیات غیرمجاز بر روی بانک اطلاعاتی انجام داده‌اند را شناسایی کند. بعضی‌اوقات از طریق اطلاعات حسابرسی می‌توان حملات آینده را نیز پیش‌بینی نمود.